EPFL

Thèse n° 8606

Gradient-based Methods for Deep Model Interpretability

Présentée le 11 novembre 2021

Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Suraj SRINIVAS

Acceptée sur proposition du jury

Prof. A. Skrivervik Favre, présidente du jury
Prof. F. Fleuret, Prof. P. Frossard, directeurs de thèse
Dr B. Kim, rapporteuse
Dr L. Denoyer, rapporteur
Prof. A. Alahi, rapporteur

École
polytechnique
fédérale
de Lausanne

2021

Abstract

In this dissertation, we propose gradient-based methods for characterizing model behaviour for the purposes of knowledge transfer and post-hoc model interpretation. Broadly, gradients capture the variation of some output feature of the model upon unit variation of an input feature, and thus encodes the local model behaviour while being agnostic to the underlying model architectural choices.

Our first contribution is to propose a sample-efficient method to mimic the behaviour of a pre-trained *teacher* model with an untrained *student* model using gradient information. We interpret our approach as an efficient alternative to data augmentation used with canonical knowledge transfer approaches, where noise is added to the inputs. We apply this to distillation and a transfer learning task, where we show improved performance for small datasets.

Our second contribution is to propose a novel saliency method to visualize the input features that are most relevant for predictions made by a given model. We first propose the *full-gradient* representation, which satisfies a property called *completeness* which provably cannot be satisfied by gradient-based saliency methods. Based on this, we propose an approximate saliency map representation called *FullGrad* which naturally captures the information within a model across feature hierarchies. Our experimental results show that FullGrad captures model behaviour better than other saliency methods.

Our final contribution is to take a step back and ask why input-gradients are informative for standard neural network models in the first place, especially when gradient structure is un-regularized and may be arbitrary. Our analysis here reveals that for a subset of gradient-based saliency maps, the map relies not on the underlying discriminative model $p(y \mid \mathbf{x})$ but on a hidden density model $p(\mathbf{x} \mid y)$ implicit within softmax-based disciminative models. Thus we find that the reason input-gradients are informative is due to the alignment of the implicit density model with that of the ground truth density, which we verify experimentally.

Keywords: Deep neural networks, knowledge transfer, distillation, saliency maps, interpretability

Résumé

Dans cette thèse, nous nous intéressons à des méthodes utilisant le gradient pour caractériser le comportement d'un modèle à des fins de transfert de connaissances et d'interprétation post-hoc des modèles. De façon générale, les gradients capturent la variation d'une caractéristique de sortie du modèle lors de la variation d'une caractéristique d'entrée, et encodent ainsi le comportement local du modèle tout en étant indépendants des choix architecturaux.

Notre première contribution est une méthode efficace (en termes du nombre d'exemples d'entraînement) pour transférer le comportement d'un modèle *enseignant* pré-entraîné vers un modèle *élève* non entraîné, en utilisant l'information du gradient. Nous voyons notre approche comme une alternative efficace à l'augmentation des données utilisée avec les approches canoniques de transfert de connaissances, dans lesquelles du bruit est ajouté aux entrées. Nous appliquons cette approche à la distillation et à une tâche d'apprentissage par transfert, où nous montrons des performances améliorées sur des ensembles de données de petites tailles.

Notre deuxième contribution est une nouvelle méthode de *saillance* pour visualiser les caractéristiques d'entrée qui sont les plus pertinentes pour les prédictions faites par un modèle donné. Nous proposons d'abord la représentation *full-gradient*, qui satisfait une propriété appelée *complétude* qui ne peut pas être satisfaite par les méthodes de saillance basées sur le gradient. Partant de ce constat, nous proposons une représentation approximative de la carte de saillance appelée *FullGrad* qui capture naturellement l'information au sein des différentes couches d'un modèle. Nos résultats expérimentaux montrent que *FullGrad* capture mieux le comportement du modèle que les autres méthodes existantes.

Notre dernière contribution consiste à analyser pourquoi les gradients d'entrée sont informatifs pour les modèles de réseaux neuronaux standards, en particulier alors que leur structure peut être arbitraire. Notre analyse révèle ici que, pour un sous-ensemble des cartes de saillance basées sur le gradient, la carte ne repose non pas sur le modèle discriminant sous-jacent $p(y \mid \mathbf{x})$ mais sur un modèle de densité caché $p(\mathbf{x} \mid y)$ implicite dans les modèles discriminants dotés d'une couche softmax. Ainsi, nous découvrons que c'est grâce à l'alignement du modèle de densité implicite avec celui de la densité réelle que les gradients d'entrée sont informatifs, propriété que nous confirmons expérimentalement.

Mots-clés : Réseaux neuronaux profonds, transfert de connaissances, distillation, cartes de saillance, interprétabilité.

Acknowledgements

This thesis is the result of the support of several people, to whom I am extremely grateful. I would first like to express my thanks and gratitude to my supervisor Prof. François Fleuret for providing the opportunity to work with him. He is truly one of the most brilliant people I have met in my life, and yet carries himself with a sense of deep humility which continues to inspire me both personally and professionally.

I would like to thank the jury members of my thesis: Prof. Pascal Frossard, Prof. Anja Skrivervik, Prof. Alexander Alahi, Dr. Been Kim and Dr. Ludovic Denoyer for their insightful questions and comments. I would also like to thank Swiss National Science Foundation for funding my research through the ISUL and CORTI projects, which helped me pursue a curiosity-driven research agenda.

My gratitude also goes to the model efficiency team at Qualcomm AI Research, particularly my mentor Tijmen Blankevoort, for giving me an internship opportunity and the freedom to explore my unusual research directions. This six-month internship at Amsterdam in the middle of the pandemic while working from home was made enjoyable due to my housemates and friends at Hotel Jansen, whom I fondly remember.

I am indebted to the administrative team at Idiap Research Institute, who made life simple for non-French speaking foreigners such as myself in helping transition to life in Switzerland. I would also like to thank the Idiap systems team for maintaining an impressive computing infrastructure, which made experimentation hassle-free.

One of the best parts of working at Idiap and EPFL was the brilliant, kind and supportive peer group that I had access to, and with whom I had the privilege of sharing many hikes, ski trips, dinners, coffee breaks and gossip sessions. I shall cherish my memories with the Amigos, the SAM-ers, the summer hikers, the skiers and snowboarders, the members of Martigny cricket club, and all the foosball players. I especially thank PJ for being my family away from family all these years, and also for tolerating my PJs. I would also like to thank the members of the Martigny Badminton Club for accepting me as one of their own, and helping me rediscover my passion for the game.

Last but not the least, I would like to thank my parents and my sister, without whose constant support and encouragement I wouldn't have been able to pursue nor persist with this outlandish dream of moving to Switzerland for my PhD.

Martigny, September 2021 Suraj Srinivas

Publications based on the Thesis

Chapter 2 of this thesis is based on:

- S. Srinivas & F. Fleuret, "Local Affine Approximations for Improving Knowledge Transfer", Workshop on Learning with Limited Data (LLD), Advances in Neural Information Processing Systems (NeurIPS), 2017 [**Best Paper Award**]
- S. Srinivas & F. Fleuret, "Knowledge Transfer with Jacobian Matching", International Conference on Machine Learning (ICML), 2018

Chapter 3 of this thesis is based on:

- S. Srinivas & F. Fleuret, "Full-Gradient Representation for Neural Network Visualization", Advances in Neural Information Processing Systems (NeurIPS), 2019

Chapter 5 of this thesis is based on:

- S. Srinivas & F. Fleuret, "Are Input-Gradients Meaningful For Interpretability?", Workshop on Human Interpretability (WHI), International Conference on Machine Learning (ICML), 2020
- S. Srinivas & F. Fleuret, "Rethinking the Role of Gradient-based Attribution Methods for Model Interpretability", International Conference on Learning Representations (ICLR), 2021 [**Oral Presentation**]

Contents

List of Figures

List of Tables

1 Introduction & Background

> One thing that connectionist networks have in common with brains is that if you open them up and peer inside, all you can see is a big pile of goo.
>
> Mozer and Smolensky (1989)

1.1 Deep Neural Networks as Black Boxes

Machine learning is a branch of artificial intelligence that studies how artificial agents can learn from experience and produce intelligent behaviours. This learning-based approach is primarily used for tasks where explicit modelling by human engineers is not possible. Recent deep learning-based approaches particularly excel at this, and these have enabled significant advances in tasks such as image classification, speech recognition and machine translation. However, the complexity of these deep models makes the internal decision-making logic within them far too complex for humans to understand, and rightfully so, as the underlying tasks themselves are complex.

However as such models continue to be applied to increasingly critical applications such as medicine, law, public policy and autonomous driving, there is a growing need to carefully understand their internal decision-making process, primarily to characterize their failure modes and understand any unintended biases they exhibit. We thus require effective tools that are able to capture the behavioural aspects of these models. Such tools can help not only domain experts involved in machine learning applications better understand the models they are working with, but these can also help researchers engaging in fundamental machine learning research, as rigorously understanding the failure modes of current models is crucial to eliminating them.

In this thesis, we study the functional behaviour of deep neural network models using gradient-based tools. In particular, we consider two broad testbeds to understand functional behaviour.

1. Knowledge Transfer: In this application, we wish to mimic the functional behaviour of a "teacher" model in another "student" model with a different architecture. Efficiently solving this problem requires having access to a useful characterization of model behaviour that can be used for learning.

2. Post-hoc Interpretability: Here, we wish to communicate model behaviour of a pre-trained model to human experts. However unlike knowledge transfer, the intent here is not for the human to mimic the model behaviour, but simply to verify its correctness.

In the rest of this chapter, we shall provide background relevant to the topics of this thesis, which we shall use to state our main contributions at the end of the chapter.

1.2 Geometry of ReLU Neural Networks

In this section, we shall introduce ReLU neural networks, which are a popular class of neural network models used in practice, and which is the main model family we consider in this thesis. A ReLU neural network is defined as the composition of functions $f(\cdot) = \mathbf{w}_n(\cdot) \circ ... \mathbf{w}_2(\cdot) \circ \sigma(\cdot) \circ \mathbf{w}_1(\cdot)$, where $\mathbf{w}_i(\cdot)$ refers to matrix multiplication with weight matrix $\mathbf{w}_i$ (with an optional bias term $\mathbf{b}_i$) and $\sigma(\cdot)$ refers to the ReLU non-linearity, which is defined as follows for some scalar x.

$$\sigma(x) = \begin{cases} 0 & x \leq 0 \\ x & \text{otherwise} \end{cases}$$

In this case, as the ReLU neural network $f(\cdot)$ consists of only linear and piecewise linear functions, the overall composition is also a piecewise linear function. Further, it can also be shown that the input region is divided into *convex* polytopes, where each polytope contains one linear piece (Montufar et al., 2014). This implies that the function is locally affine almost everywhere, and the local Taylor series expansions are exact in a small neighbourhood around an input $\mathbf{x} \in \mathbb{R}^d$.

$$f(\mathbf{x} + \epsilon) = f(\mathbf{x}) + \epsilon^\top \nabla_{\mathbf{x}} f(\mathbf{x}) \ ; \ \text{for } \|\epsilon\|_2 \leq M$$

In other words, there exists a ball of some radius $M > 0$ (which is itself a function of f and $\mathbf{x}$) such that the above Taylor expansion is exact. This also implies that the Hessian of the model w.r.t. input is either zero almost everywhere and undefined at the boundary between pieces. These facts imply that the local neighbourhood of ReLU neural networks is characterized completely by the gradient term $\nabla_{\mathbf{x}} f(\mathbf{x})$, which motivates its use for our case.

Further, consider the special case of ReLU neural networks without bias, i.e., $\mathbf{b}_i = 0$. Such models have a special property called non-negative homogeneity, which implies that $f(k\mathbf{x}) = kf(\mathbf{x})$ for any non-negative scalar $k \geq 0$. This is due to the fact that both matrix multiplication with $\mathbf{w}_i$ and the ReLU non-linearity share this property, and as such their composition also retains this property. Now let $\epsilon \in \mathbb{R}^+$ be a small positive scalar. We can now use first-order Taylor series to write the following. $f((1+\epsilon)\mathbf{x}) = f(\mathbf{x}) + \epsilon f(\mathbf{x}) = f(\mathbf{x}) + \epsilon \mathbf{x}^T \nabla_{\mathbf{x}} f(\mathbf{x})$. Using this we have

$$f(\mathbf{x}) = \mathbf{x}^\top \nabla_{\mathbf{x}} f(\mathbf{x})$$

This implies that as a result of the non-negative homogeneity property, the output of the model can be written purely in terms of the input, and the gradients of the output w.r.t. input. Thus in this case, the gradient conveys all the information required to compute the model output in a parameterization independent manner. We shall call this property of gradients as *completeness*, as it completely captures all information required to compute the function output.

Thus while the gradient is a powerful tool in general for the analysis of non-linear models, it seems particularly relevant for the case of ReLU neural networks because of the two properties discussed above: piecewise linearity and completeness in case of zero-bias models.

In the next section, we shall introduce the task of knowledge transfer in the context of our contributions, and then proceed to review relevant literature.

1.3 Knowledge Transfer Between Deep Models

Assume we have a (teacher) model f trained on a particular dataset. We would now like to train another (student) model g with a different architecture on the same, or a related dataset. Is it possible to leverage f to train g more efficiently? This is the research question which knowledge transfer aims to address. Distillation (Buciluǎ et al., 2006) is a form of knowledge transfer where f and g are trained on the same dataset, but have different architectures. Transfer Learning (Pan and Yang, 2010) is another form of

knowledge transfer where f and g are trained on different (but related) datasets. If the architectures are the same, we can in both cases simply copy weights from f to g. The problem becomes more challenging when f and g have different architectures, which is exactly the problem we wish to solve. In other words, we have that $f \in \mathcal{F}$ and $g \in \mathcal{G}$ belong to different model classes, such that $g \notin \mathcal{F}$.

A common strategy to solve this problem is to align the underlying functions themselves by minimizing some pointwise distance $\mathcal{D}$ between function outputs for some set of inputs $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N\}$:

$$\min_{g \in \mathcal{G}} \mathbb{E}_{\mathbf{x} \in \mathcal{X}} \mathcal{D}(f(\mathbf{x}), g(\mathbf{x}))$$

Common choices for the distance function $\mathcal{D}$ include mean-squared error (Ba and Caruana, 2014) or the cross-entropy loss between softmax-normalized outputs (Hinton et al., 2015). Often, this strategy alone is insufficient, and can be improved upon by assuming some similarities in the architectures between f and g. For instance, if we assume that both models are deep and convolutional, then it is possible to align intermediate representations (Romero et al., 2014; Zagoruyko and Komodakis, 2017) in addition to aligning function outputs.

One drawback of such distillation approaches is that they often require large datasets to be able to accurately mimic the teacher. One way to work around this problem for small datasets is to use data augmentation to increase the size of the dataset. While these are effective, these require domain knowledge to design good data augmentation schemes. If uninformed random inputs are used for distillation, then the student model performance on real data suffers, as mimicing is performed on a subset of data not relevant for the task. Thus the usage of good data augmentation schemes is crucial.

In Chapter 2, we shall propose a distillation procedure for small datasets based on matching gradients of the input-output map, which we show is equivalent to performing data augmentation, where random noise is added to the inputs. In the rest of the section we shall provide more information about such input-gradients, especially for ReLU neural networks.

1.3.1 Related Work on Knowledge Transfer

In this thesis, we combine two well-known areas of transfer learning and distillation into one broad term of knowledge transfer. Broadly, distillation involves transferring representations from one model to another for the same dataset, and transfer learning involves transferring the representations learnt for one task to use for another task. In

this part, we shall review literature to both distillation and transfer learning approaches that are related to distillation.

Knowledge Distillation

The study of distillation is commonly viewed via the lens of model compression (Buciluǎ et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015), where one is required to train student models that are smaller than pre-trained teacher models. Such student models can be designed to be smaller either in terms of number of parameters, number of FLOPs, or run-time taking into account hardware constraints (Gou et al., 2021). An example of such a use-case is in the area of speech synthesis (Oord et al., 2018), where the teacher model is a highly performant but slow autoregressive model, and the student is a typically less accurate but fast convolutional model. Using techniques of distillation, Oord et al. (2018) were able to improve the accuracy of the student convolutional model, which resulted in a synthesis speed-up of several orders of magnitude. Canonical distillation approaches train the student with a linear combination of two loss functions, one encouraging the student to mimic the teacher with either a cross-entropy or a mean-squared error loss, and the other encouraging the student to fit the ground truth labels hard labels well. The soft targets produced by teacher models in this case are typically more informative than hard target labels, which is one explanation for the success of these approaches.

Feature-matching for Distillation

The standard distillation approach can be improved upon by incorporating additional information from the teacher model, thus providing the student with stronger supervision for learning, which can potentially improve learning outcomes. One possible way to do this would be to match intermediate features from a teacher to a student model. However, one problem here is that feature dimensionality (width) and number of features (depth) may be different from the student and teacher models. To overcome the difference in depths, a common approach is to only match layers whose width is most similar across the two models. To overcome difference in width, there are different strategies employed in practice. While Romero et al. (2014) use a fully connected or 1×1 convolutional adaptor layer to match the number of channels and spatial resizing to match the spatial dimensions, (Zagoruyko and Komodakis, 2017) collapse the channel dimension in convolutional layers using an aggregation function and match only the aggregated map. Such aggregated feature maps are termed "attention" maps, and thus make a compelling connection between interpretability and distillation, that of using saliency map techniques for improving distillation performance. In Chapters 2 and 4, we use input-gradients and full-gradients, which are interpretability methods, for distillation and find that these improve distillation performance. Another strategy for aggregating

information involves computing a correlation matrix between two layers with activations which have the same spatial size, thus resulting in a matrix whose dimensions are equal to the widths or number of channels of the two layers involved (Yim et al., 2017). Then proceeds to match this quantity between two models. This procedure can be applied to early and later layers of a resnet block for instance, and the correlation matrix, called the FSP matrix (Yim et al., 2017) captures the change in information within such layers.

Data-free Distillation

One of the main problems with knowledge distillation approaches is that they require access to large datasets to obtain a well performing student model. However, there may exist situations where one may not have access to the dataset for privacy reasons, or if the dataset is too large to store. The straightforward approach in this scenario is to simply perform distillation with random data points instead of points from the dataset. However matching on random points leads to imperfect matching on dataset points and thus leads to suboptimal performance (Buciluǎ et al., 2006). It is instead preferable to build a class-conditional or joint density model of the dataset and then use samples from this density model for distillation. While early approaches to this use simple heuristics to increase dataset size (Buciluǎ et al., 2006), modern approaches typically involve using GANs to generate samples (Yoo et al., 2019; Micaelli and Storkey, 2019). While (Yoo et al., 2019) train a class conditional Generative Adversarial Networks (GAN) to approximate the dataset directly, (Micaelli and Storkey, 2019) instead use GANs to approximate the density of samples where the teacher and student predictions disagree the most, and then improve distillation on these adversarial samples. In contrast, (Nayak et al., 2019; Yin et al., 2020) also use an implied generative model using only the teacher model and without using additional models such as GANs. In particular, the implied generative model involves optimizing the input such that either softmax outputs agree with dataset statistics (Nayak et al., 2019), or such that the batchnorm statistics match that of the dataset (Yin et al., 2020). Using these procedures they are able to generate data for distillation purposes. However, it is unclear whether this avoids problems with privacy, especially when generative models can still leak sensitive information learnt from the data, but can still be useful in avoiding to explicitly store datasets.

Self-Distillation

In most applications involving distillation, the student model is simpler than the teacher. However, in a form of distillation called *self-distillation*, the teacher and student models are identical. The motivation for this form of distillation is no longer a transfer of representations, which is trivial to perform in this case, but to improve optimization of the teacher model. In Furlanello et al. (2018), it was found that if a student model is iteratively distilled on previously obtained teacher models, then the performance of the

student are found to be better than the teachers. In this case, the terminology "teacher" and "student" is made to relate to the distillation literature, however the algorithm itself is reminiscent of functional optimization strategies such as natural gradient descent which ensure that successive optimization steps are close in function space as opposed to parameter space. A related approach by Xie et al. (2020) proposed usage of noise addition to the student during training, which was found to further improve performance. Other forms of self distillation in literature include approaches that distill information from early layers of the model to later ones (Phuong and Lampert, 2019b; Zhang et al., 2019), and those that distill information from early epochs to later ones (Yang et al., 2019) in a manner analogous to Furlanello et al. (2018).

Theory of Distillation

The empirical success of distillation methods still leave an unanswered question of what factors cause this success, and why soft labels used in distillation lead to efficient learning. Phuong and Lampert (2019a) analyze this problem for the case of single-layer and deep linear models, and derive generalization bounds for (self) distillation in this case. They find here that the important factors that determine learning efficiency are data geometry, optimization bias, and size of the dataset which results in a property called strong monotonicity. Mobahi et al. (2020) also analyze the self-distillation problem for non-linear models in hilbert space, and show that self-distillation procedure acts as a form of regularization and restricts the number of basis vectors that can be used represent the solution. Cho and Hariharan (2019) empirically show that more accurate teacher models do not necessarily lead to better student models, and that a mis-match between the capacities of these models leads to worse performance. They also suggest one way to guard against this phenomenon by performing early stopping during the training of the teacher which restricts the complexity of the learnt hypothesis.

Transfer Learning and Continual Learning

Related to distillation is the task of transfer learning which involves transferring representations learnt on one task A to another task B. Another related task is that of continual learning, which requires that the transfer happen such that the representation still be suitable for both tasks, whereas transfer learning only cares about performance on task B. In both cases, we assume that samples used to learn task A are unavoidable when transferring to task B. The canonical method for transfer learning is fine-tuning, which involves learning on task B with a model initialized with the learnt weights of another model trained on task A. However, one problem with fine-tuning is that during the course of learning the model could diverge significantly from the representations of task A, thus reducing performance on task B. Rozantsev et al. (2018) propose to tackle this problem by penalizing deviations from the parameters learnt for task A while learn-

ing to solve task B. While this penalizes deviations in parameter space, it is preferred to directly penalize deviations in function space, as the two often do not coincide especially for the deep neural networks where parameter space is degenerate. To alleviate this, Kirkpatrick et al. (2017); Zenke et al. (2017) propose to penalize deviations in function space by using local quadratic approximations to the function deviation which results in the usage of the Fisher information matrix or the Hessian. In contrast, the LwF method (Li and Hoiem, 2016) directly penalizes deviation in function space using an explicit distillation objective, which thus connects distillation to the auxiliary tasks of transfer learning and continual learning. LwF performs transfer learning by learning a student model which mimics the response of a teacher model trained for task A, on samples used to learn task B. This is done in addition to learning task B itself. While the first objective transfers representations from task A, the second objective learns task B, thus enabling transfer learning. When the student model is initialized from the teacher, this enables continual learning, as it requires the model's representations change very little from the initialization. Rebuffi et al. (2017) use a similar principle for class-incremental learning, where examples from individual classes constitute tasks. However, in addition, they also build and update a small exemplar set of examples from previous tasks which aids learning.

This concludes our discussion on knowledge transfer. In the next section, we shall introduce post-hoc interpretation of neural networks, discuss our contributions and similarly review literature on the broader area of interpretability.

1.4 Post-hoc Interpretability of Machine Learning Models

We are often required to explain the internal decision-making process of machine learning models to human experts such that they are able to effectively interface with such models. These are called *post-hoc* interpretations, where model behaviour is explained after training. In this context, we are required to communicate model behaviour to an expert in a human-friendly manner. One such human-friendly modality of explanation is that of feature importance methods, which provides importance scores to each input feature of the model. Ideally, the magnitude of the importance value indicates the extent to which the model relies on a particular feature, with high importance indicating high reliance and zero importance indicating no reliance on that feature. However, there are many ways to define such importance scores and this is an active topic of research. In particular, each definition of importance leads to a different feature importance approach, thus explaining the plethora of such methods in literature. Here we shall review two popular feature importance methods based on two different notions of importance.

1.4.1 Input-Gradients: Feature Importance via Sensitivity

One popular method to define feature importance is based sensitivity to perturbation of inputs. In this case, a feature is deemed more important than another if the model is highly sensitive to random perturbations to that feature over another. This definition of feature importance is called *sensitivity*. When the noise added is infinitesimally small, this sensitivity to perturbations is exactly captured by the magnitude of the input-gradients of the model. More formally, for a model f, we have the feature importance map, or the *saliency* map $\mathcal{S}(f, \mathbf{x})$ defined as follows (Simonyan et al., 2013).

$$\mathcal{S}(f, \mathbf{x}) = |\nabla_{\mathbf{x}} f(\mathbf{x})|$$

While this is an appealing method to define feature importance, this suffers from the problem of *non-attribution during saturation* (Shrikumar et al., 2017). In other words, consider some $\mathbf{x}$ at the local extrema, saddle points, or flat regions of f. For these points, the gradient is always equal to zero and thus all features are assigned zero importance. While this agrees with the definition of sensitivity, it is incorrect and misleading to state that model output does not rely on any input feature, as the output is explicitly a function of the input. Another problem with using sensitivity to small perturbations is that model behaviour can change drastically over larger noise scales, thus diverging from the importance scores provided by input-gradients.

A straightforward solution to both these problems is to simply measure sensitivity to large noise scales directly, for instance by measuring the input-Hessian. However for these we have an additional problem of having to account for the joint effects of perturbing multiple features together. In other words, while the gradient has only N terms for an N-dimensional input, the Hessian has $\mathcal{O}(N^2)$ terms, and in general for larger scales we need to account for $\mathcal{O}(N^N)$ interactions terms, which are intractable to compute and visualize. This means that in general we need to measure not just importances of individual features, but also of collections of features together, if we want to completely characterize model behaviour at larger noise scales.

1.4.2 Integrated Gradients: Feature Importance via Completeness

A different principle to define feature importance is based on the idea of *completeness*. In this view, each input feature is assumed to have an contribution towards producing the output such that the sum of contributions of each input feature equals that of the model output. To formalize this idea, the principle of Shapley values from economics is often used (Lundberg and Lee, 2017; Sundararajan et al., 2017), where an analogy is drawn to N-player co-operative games, where any reward received by the team as a whole must be

re-distributed to individual players in a 'fair' manner. The considerations for such fair re-distribution are (a) a weak notion of dependence, where if a variable (player) does not contribute mathematically, then its attribution is zero (b) linearity-preservation, i.e., a linear combination of functions (teams) leads to a similar linear combination of their attributions of each player, (c) symmetry preservation, which states that if a function is symmetric w.r.t. two variables (players), then their attributions are equal. These considerations are shown to lead to a saliency method termed as *integrated gradients* (Sundararajan et al., 2017), which are defined as follows for the i^{th} input feature, and for a fixed baseline input $\mathbf{x}'$.

$$\mathcal{S}(f, \mathbf{x}, \mathbf{x}')_i = (\mathbf{x}_i - \mathbf{x}'_i) \times \int_{\alpha=0}^{\alpha=1} \frac{\partial f_i(\mathbf{x}' + \alpha(\mathbf{x} - \mathbf{x}'))}{\partial \mathbf{x}_i}$$

Note that from the fundamental theorem of calculus, the following holds

$$\sum_i \mathcal{S}(f, \mathbf{x}, \mathbf{x}')_i = f(\mathbf{x}) - f(\mathbf{x}')$$

Here, $\mathbf{x}'$ is a so-called reference or baseline input representing an input with 'zero' signal. Often this can be chosen to be zero, but can also be for instance, mean or noise input. The property above is called *completeness* with a baseline.

This method solves the problem of non-attribution during saturation, as a non-zero model output always results in non-zero attributions irrespective of the input being at a local extrema of the function. However, this now has the problem that the dependency condition it satisfies is very weak, in that it only implies that if a function that does not depend mathematically on a variable for all inputs, then its attribution is zero. This means that if one feature is assigned a higher importance than another, the precise meaning of this is unclear, as this score is given based on multiple considerations for a 'fair' re-distribution, not a single simple principle like sensitivity. In particular for integrated gradients, the score for some i^{th} feature **cannot** be interpreted as the change in model output upon changing the i^{th} feature from $\mathbf{x}_i$ to $\mathbf{x}'_i$ (or vice versa), as the score in this case is computed by changing all features together, and not individual features in isolation. For linear models, there is no interaction between features and thus this interpretation is valid for this case, however this is not true for more general non-linear models.

Overall, feature importance methods are based on defining importance scores to each input feature, whereas commonly used non-linear models such as deep models have a large number of feature-wise interactions, limiting the utility of such methods in practice.

In Chapter 3, we shall define a representation which captures both completeness and a strong notion of sensitivity, and naturally captures importance of not just individual features, but that of patches at different scales for convolutional models.

1.4.3 Related Work on Interpretability

While the topic of this thesis focusses on post-hoc interpretability, in this part we shall briefly discuss the broader literature around interpretability and how post-hoc interpretability relates to the rest of the field.

Definition and Goals of Interpretability

The goals of interpretability in machine learning are to be able to perform human introspection for models for a wide variety of use cases such as debugging, reasoning about model behaviour in unknown settings for safety, verifying fairness claims, the ability to audit model decisions and reasoning about mis-matched objectives (Doshi-Velez and Kim, 2017; Weller, 2019). Note that these goals are distinct from one another, and as such, methods that achieve one goal may not achieve others. For instance, existing work in the fairness literature provides methods to achieve model fairness (Barocas et al., 2019), however these do not necessarily fall under the purview of interpretable machine learning as they do not help with the other goals of interpretability. To further clarify what constitutes interpretability, Lipton (2018) make a classification into three types of interpretability, namely, *simulatability*, *decomposability*, and *algorithmic transparency*. While simulatability refers to being able to simulate the model computations by humans, decomposability refers to each part of the model and the input being simulable, and algorithmic transparency refers to the overall learning algorithm being well understood analytically. Thus these different requirements might lead to different models that may be interpretable with respect to one definition but not another. For example, Lipton (2018) argue that linear models are not interpretable for high dimensional inputs when it comes to simulatability or decomposability, but are so when considering algorithmic transparency. On the other hand, post-hoc interpretability approaches which assign concept labels to intermediate neurons within neural network models may render them interpretable w.r.t. decomposability, but not other notions. Thus it is important that interpretability approaches clearly specify under which sense they are interpretable.

Interpretable Model Families

One challenge in interpretable machine learning is the identification of model families that are interpretable according to one of the senses described above. Note that the model families that are interpretable may be distinct for each use case. For instance, linear models are typically considered to be interpretable due to the simplicity of the

Model Family	Application
decision trees / sets	clean tabular data with interactions, including multiclass problems. Particularly useful for categorical data with complex interactions (i.e., more than quadratic)
scoring systems (sparse linear models with integer co-efficients)	somewhat clean tabular data, typically used in medicine and criminal justice because they are small enough that they can be memorized by humans.
generalized additive models (GAMs)	continuous data with at most quadratic interactions, useful for large-scale medical record data.
case-based reasoning	any data type (different methods exist for different data types), including multiclass problems.
disentangled neural networks	data with raw inputs (computer vision, time series, textual data), suitable for multiclass problems.

Table 1.1 – General rules of thumb for interpretable model families that apply to different use cases. Table from Rudin et al. (2021).

model, however as discussed above, this may not be true for high-dimensional inputs (Lipton, 2018). Setting aside such special cases, Rudin et al. (2021) specify rules of thumb for what models classes are interpretable for different applications, which we show in Table 1.1. In the table, decision trees, decision lists and decision sets are logical models that process inputs using if-else type rules. While decision trees and lists typically check whether certain constraints are satisfied between the inputs, decision sets use or-and type rules to process inputs. These are models typically applicable to tabular data, and are interpretable for small tree sizes. Other models include scoring systems that are sparse linear models with integer co-efficients that are also applicable in similar domains. Generalized Additive Models (GAMs) (Hastie and Tibshirani, 1986) are a class of models that process each input co-ordinate separately using non-linear transformations, and are thus still interpretable due to the absence of across feature interactions. These have also been used with neural network features, in a model class called 'neural additive model' (Agarwal et al., 2020). Similar approaches which process images patch-wise have also shown promise for large scale tasks such as image classification despite the restrictive nature of the model (Brendel and Bethge, 2019). Case-based reasoning refers to a general strategy of modelling where predictions are made based on exemplars in the training set, examples being nearest neighbour and kernel-based classifiers. Neural network models that use such case-based reasoning have also been used shown to achieve good performance on large scale tasks (Li et al., 2018; Chen et al., 2019). Finally, disentangled neural networks refer to models where the hidden neurons

represent one semantic concept as opposed to a mixture of them. However encouraging such disentanglement has been show to adversely affect model performance Leavitt and Morcos (2021). Overall, once we have identified the correct model class, the only remaining challenge is to be able to train these models well. However, given that these model classes are often less expressive when compared to their non-interpretable counterparts, predictive performance is likely to suffer.

Post-hoc Interpretability

Post-hoc interpretability involves explaining existing pre-trained models. This is convenient, as it does not require model re-training, and interpretation can be achieved without sacrificing model performance. However, this assumes that such pre-trained models are amenable to interpretation in the first place, which need not strictly be true. Some common approaches to post-hoc interpretability include, feature attribution methods, and case-based reasoning methods. Feature attribution methods or local explanation methods assign importance scores to each input feature typically in a local neighbourhood. A canonical approach here is to approximate the model locally using a linear model (Ribeiro et al., 2016). Lundberg and Lee (2017) extend and generalize this to the class of additive models and propose SHAP, which uses Shapley values to assign feature importance. Gradient-based saliency methods (Simonyan et al., 2013; Smilkov et al., 2017; Sundararajan et al., 2017; Zeiler and Fergus, 2014; Springenberg et al., 2014; Montavon et al., 2017; Shrikumar et al., 2017; Selvaraju et al., 2017) on the other hand, define feature importance using criteria such as sensitivity or completeness, which was described in the previous sections. There also exist non-gradient based methods (Zintgraf et al., 2017; Chang et al., 2019) which measure importance using input perturbation. Case-based reasoning is another general explanation paradigm for explaining individual predictions by relating them to predictions made on other known points, such as those in the training set. Canonical approaches here include the use of influence functions (Koh and Liang, 2017), which estimate the effect of each training sample on the final test prediction, using which they are able to obtain critical training examples for prediction.

Limitations of Post-hoc Interpretability

One limitation of post-hoc interpretability is that the explanation mechanism is distinct from the model, and in general the two can diverge significantly (Rudin, 2019). This particularly holds true in the realm of gradient-based feature importance methods applied to deep neural networks, as these models are typically highly non-linear and thus gradients are only applicable in a small neighbourhood. As a result, it is important to quantitatively evaluate such saliency maps to ensure that they do indeed capture meaningful aspects of the model. A common test in this regime is the pixel perturba-

tion test (Samek et al., 2016), which consists of perturbing the least (or most) salient pixels according to a saliency map, by replacing them with zero information pixels and measuring the resulting change in model output. A good saliency map is that which shows smaller output change upon perturbing less important pixels and vice verse for important ones. Another such test was proposed by Adebayo et al. (2018) who propose to measure the variation of the saliency map to model randomization. In particular, it is expected that the saliency map must be random upon model randomization. However, Adebayo et al. (2018) showed that several gradient-based saliency methods are invariant to such randomization. This shows that saliency map structure in these cases depends on statistical properties of the model weights and not the input-output map itself, which is an undesirable property. This was proven rigorously (Nie et al., 2018) for the case of saliency maps such as guided backprop and deconvolution.

Orthogonal to this, there has also been work showing that saliency map explanations are easy to fool. In particular, Ghorbani et al. (2019); Zhang et al. (2020) show that it is possible to find "adversarial" images close to the original images such that while the predicted label remains the same, the saliency map is vastly different. Similar observations are made by Dombrowski et al. (2019) who show that this is possible because of the geometry of neural network models, and in particular their large curvature. Similarly, Subramanya et al. (2019) show that it is possible to fool both saliency maps and predictions by simply imperceptibly modifying the same image patch for all images. Distinct from this, Heo et al. (2019) show that it is possible to modify the model parameters, such that model prediction is still accurate but the explanations are arbitrary. We expand on this problem and provide an explanation for this phenomenon in Chapter 5 based on the shift-invariance of softmax. Overall, the susceptibility of neural network models to adversarial manipulation indicates that these maps may not capture model behaviour in all cases.

In the next section, we shall discuss an alternative approach to knowledge transfer and interpretability, based on density models implicit within standard discriminative ones.

1.5 Density Modelling via Discriminative Models

For both knowledge transfer and interpretability, we require tools that capture the knowledge encoded within neural network models. One set of tools that has shows promise for this purpose are generative models embedded within discriminative ones. Such tools have been used for zero-shot knowledge transfer (Nayak et al., 2019; Yin et al., 2020; Haroush et al., 2020), where data to perform knowledge transfer is generated from the 'teacher' model using a heuristic data-generating process usually involving optimization of inputs to maximize certain neuronal activations. This generated data is fed to the 'student' after which usual knowledge transfer methods are applied. While these methods do not consider these data-generating process as sampling methods from an explicit

generative model, we observe that the procedures used bear a striking resemblance to sampling from energy-based generative models (EBMs). In Chapter 5, we consider an energy-based generative modelling interpretation for softmax-based generative models and connect this to the interpretability properties of input-gradients. In the rest of this section, we shall provide the requisite background on EBMs.

Given a set of data points $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^D\}$ from some distribution $p_{\text{data}}(\mathbf{x})$, the objective of generative modelling is to recover $p_{\text{data}}(\mathbf{x})$ using only the samples $\mathcal{X}$. Energy-based generative models are generative models parameterized by an un-normalized energy function h, as follows:

$$p_{\text{ebm}}(\mathbf{x}) = \frac{\exp\left(h(\mathbf{x})\right)}{\int_{\mathbf{x}'} \exp\left(h(\mathbf{x}')\right) \mathrm{d}\mathbf{x}'}$$

Thus we define a density function $p_{\text{ebm}}(\mathbf{x})$ indirectly by defining an arbitrary function h which constitutes the numerator. The denominator is obtained by integrating this quantity over the input domain, and is called the partition function. The objective of generative modelling in this case is to find h such that $p_{\text{data}}(\mathbf{x}) \approx p_{\text{ebm}}(\mathbf{x})$. In the sections below we shall discuss more about how to train and sample from such energy based models.

1.5.1 Sampling from EBMs

The general procedure to sample from EBMs is to use Markov Chain Monte Carlo (MCMC) procedures, which are designed to sample without computing the partition function. Algorithms in this family include the classical Metropolis-Hastings algorithm (Hastings, 1970). However this typically has a large 'burn-in' time, i.e., time taken to reach a high density region from low density ones, that scales with dimensionality, which makes them impractical for use with high-dimensional densities. An alternate approach is to use Langevin Dynamics (Neal et al., 2011), which roughly performs noisy gradient ascent on the density model for $i \in [1, K]$ steps, as follows

$$\mathbf{x}_0 \sim p_0(\mathbf{x}), \quad \mathbf{x}_{i+1} \leftarrow \mathbf{x}_i - \epsilon \nabla_{\mathbf{x}} p_{\text{ebm}}(\mathbf{x}_i) + \sqrt{2\epsilon}\, \mathbf{z}_i, \quad \mathbf{z}_i \sim \mathcal{N}(0, \mathrm{I})$$

Here $p_0(\mathbf{x})$ is some arbitrary initial distribution, a common choice for which is the uniform distribution over the input domain. When $\epsilon \to 0$ as number of steps $K \to \infty$, this procedure converges to samples from $p(\mathbf{x})$ (Neal et al., 2011). However usually in practice, a fixed ϵ is chosen and noise is added independent of that resulting in a

biased sampler (Grathwohl et al., 2020). When no noise is added to the step above, the algorithm recovers the mode of the density $p_{\text{ebm}}(\mathbf{x})$ rather than samples.

1.5.2 Training EBMs

While there are a multitude of methods to train EBMs, we shall here focus on the canonical approach, that being approximate maximum likelihood. Here we wish to maximize the likelihood of $p_{\text{ebm}}(\mathbf{x})$ on the data samples $\mathcal{X}$. Assume that the energy function in this case has parameters θ. We can now compute the gradient of the log density w.r.t. these parameters, which is usually approximated with the following expression.

$$\nabla_\theta \log p_{\text{ebm}}(\mathbf{x};\theta) \approx \mathbb{E}_{p_{\text{ebm}}(\mathbf{x}';\theta)} \nabla_\theta h(\mathbf{x}';\theta) - \nabla_\theta h(\mathbf{x};\theta)$$

In order to compute the expectation in the expression above, we require MCMC based approaches such as Langevin Dynamics to sample from p_{ebm}, which can make training slow.

In Chapter 5 we shall describe an alternate method of training EBMs, called score-matching, which does not require use of such MCMC procedures at train time.

1.6 Research Questions & Contributions

Having discussed the relevant background material, we are now ready to state the concrete research questions we consider in this thesis.

Research Question 1 *Is it possible to perform efficient knowledge transfer in a parameterization-independent manner?*

We answer this in the affirmative in Chapter 2 by proposing the use of the gradients of the input-output map for knowledge transfer. Here we first establish an equivalence between gradient matching and a data augmentation procedure, where noise is added to the inputs. We then rely on this analysis to apply gradient matching to transfer learning by establishing equivalence of a recent transfer learning procedure to distillation. We show experimentally on standard image datasets that gradient-based penalties improve distillation, robustness to noisy inputs, and transfer learning.

Research Question 2 *Is it possible to define feature importance maps that satisfy both 'sensitivity' and 'completeness' properties?*

We answer this in the negative in Chapter 3, where we show that no feature importance map can satisfy both these properties at once. We then define an alternate representation called the *full-gradient* representation that is more expressive than feature importance maps, and captures both properties simultaneously. Based on this representation, we also define an approximate saliency map representation for convolutional nets, called *FullGrad* which is obtained by aggregating gradient information across layers, and thus naturally captures the hierarchical nature of computation. Experiments show that *FullGrad* captures model behaviour better than other saliency methods in literature. In addition, we also show that the full-gradient representation can improve sample complexity of distillation, and can also be used to regularize intermediate layers of neural networks, which we show in Chapter 4.

Research Question 3 *Why are gradient-based feature importance maps successful in capturing model behaviour?*

In Chapter 5 we take a bird's eye view on gradient-based feature importance methods, and ask what makes them successful in predicting model discriminiative behaviour $p_\theta(y \mid \mathbf{x})$ in the first place. We start with the observation that commonly used saliency maps which use the gradients of the logits w.r.t. input can be arbitrary due to the shift-invariance of softmax. This leaves an open question of why these gradients are structured when in fact they can be completely arbitrary. To resolve this, we re-interpret logits of softmax-based discriminative models as unnormalized energy functions of the underlying data distribution, and show that logit-gradients are gradients of this class-conditional generative model $p_\theta(\mathbf{x} \mid y)$. We hypothesize that the interpretability properties of logit-gradients relate to the alignment of this latent generative model to the ground truth $p_{\text{data}}(\mathbf{x} \mid y)$, which we show via experiments. Overall, this work shows two things, (1) logit-gradients capture properties of a implicit generative model, and hence we must rethinking their use for interpreting discriminative models, (2) the structure of such gradient based maps relies on the alignment of the implicit generative model with the ground truth, which is independent of the properties of the discriminative model.

1.7 Notations

Symbol	Description
$\mathbf{x}, y$	Input data ($\mathbf{x}$) and corresponding output label (y) for classification tasks
$\mathcal{X}$	Dataset consisting of N input-output data pairs, i.e., $\mathcal{X} = \{(\mathbf{x}_i, y_i); i \in [1, N]\}$
D	Dimensionality of $\mathbf{x}$, i.e., $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$
N	# points in dataset $\mathcal{X}$
f, g	Neural Network functions
θ, ϕ	Vector of parameters for the models f, g respectively
$p_\theta(y \mid \mathbf{x})$	Probabilistic discriminative model with parameters θ
$p_\theta(\mathbf{x} \mid y)$	Class-conditional density model of $\mathbf{x}$ given y with parameters θ
$p_{\text{data}}(\mathbf{x})$	Ground truth density model of input $\mathbf{x}$

Table 1.2 – Notations used in the thesis.

2 Knowledge Transfer with Jacobian Matching

Classical distillation methods transfer representations from a "teacher" neural network to a "student" network by matching their output activations. Recent methods also match their Jacobians, or the gradient of output activations with the input. However, this involves making some ad hoc decisions, in particular, the choice of the loss function. In this chapter, we first establish an equivalence between Jacobian matching and distillation with input noise, from which we derive appropriate loss functions for Jacobian matching. We then rely on this analysis to apply Jacobian matching to transfer learning by establishing equivalence of a recent transfer learning procedure to distillation. We then show experimentally on standard image datasets that Jacobian-based penalties improve distillation, robustness to noisy inputs, and transfer learning.

2.1 Introduction

Consider that we are given a neural network $\mathcal{A}$ trained on a particular dataset, and want to train another neural network $\mathcal{B}$ on a similar (or related) dataset. Is it possible to leverage $\mathcal{A}$ to train $\mathcal{B}$ more efficiently? We call this the problem of *knowledge transfer*. Distillation (Hinton et al., 2015) is a form of knowledge transfer where $\mathcal{A}$ and $\mathcal{B}$ are trained on the same dataset, but have different architectures. Transfer Learning (Pan and Yang, 2010) is another form of knowledge transfer where $\mathcal{A}$ and $\mathcal{B}$ are trained on different (but related) datasets. If the architectures are the same, we can in both cases simply copy weights from $\mathcal{A}$ to $\mathcal{B}$. The problem becomes more challenging when $\mathcal{A}$ and $\mathcal{B}$ have different architectures.

A perfect distillation method would enable us to easily transform one neural network architecture into another, while preserving generalization. This capability would allow us to easily explore the space of neural network architectures, which can be used for neural network architecture search, model compression, or creating diverse ensembles. A perfect transfer learning method, on the other hand, would use little data to train $\mathcal{B}$,

optimally using the limited samples at its disposal.

This chapter deals with improving knowledge transfer by matching the Jacobians of the networks' outputs with respect to their inputs. This approach has also been recently explored for the case of distillation by Czarnecki et al. (2017), who considered the general idea of matching gradients, and by Zagoruyko and Komodakis (2017) who viewed gradients as attention maps. However it was unclear how these methods were related to classical distillation approaches (Ba and Caruana, 2014; Hinton et al., 2015), making it difficult to identify reasons for improved performance.

Recently Li and Hoiem (2016) proposed a distillation-like approach to perform transfer learning. However its precise relationship to distillation was unclear, making it difficult to predict whether improvements in distillation would lead to improvements in transfer learning.

The overall contributions of this chapter are:

1. We show that matching Jacobians is a special case of classical distillation, where noise is added to the inputs.

2. We show that a recent transfer learning method (LwF by Li and Hoiem, 2016) can be viewed as distillation, which allows us to match Jacobians for this case.

3. We provide methods to match Jacobians of practical deep networks, where architecture of both networks are arbitrary.

We experimentally validate these results by providing evidence that Jacobian matching helps both distillation and transfer learning, and that gradient-norm penalties can be used to learn models robust to noise.

2.2 Related Work

Several gradient-based regularizers have been proposed in literature. Sobolev training (Czarnecki et al., 2017; Wang et al., 2016), showed that using higher order derivatives along with the targets can help in training with less data. This work is similar to ours. While we also make similar claims, we clarify the relationship of this method with regular distillation based on matching activations. Specifically, we show how specifying the loss function used for activation matching also specifies the loss function for Jacobian matching. Zagoruyko and Komodakis (2017) introduced the idea of matching attention maps, of which gradients were an instance. This work found that combining both activation matching and gradient matching was helpful, which is a natural consequence of analysis in our work.

Drucker and Le Cun (1992) considered penalizing the gradient norm of neural networks. The intuition was to make the model more robust to small changes in the input. We find that this conforms to our analysis as well.

Knowledge Distillation (Hinton et al., 2015) first showed that one can use softmax with temperature to perform knowledge transfer with neural nets. Ba and Caruana (2014) found that squared error between logits worked better than the softmax method, and they used this method to train shallow nets with equivalent performance to deep nets. Romero et al. (2014) and Zagoruyko and Komodakis (2017) showed how to enhance distillation by matching intermediate features along with the outputs, but used different methods to do so. Sau and Balasubramanian (2016) found that adding noise to logits helps during teacher-student training. We show that the use of the gradient can be interpreted as adding such noise to the inputs analytically.

2.3 Jacobians of Neural Networks

Let us consider the first order Taylor series expansion of a function $f : \mathbb{R}^D \to \mathbb{R}$ around a small neighborhood $\{\mathbf{x} + \Delta\mathbf{x} : \|\Delta\mathbf{x}\| \leq \epsilon\}$. It can be written as

$$f(\mathbf{x} + \Delta\mathbf{x}) = f(\mathbf{x}) + \nabla_x f(\mathbf{x})^T(\Delta\mathbf{x}) + \mathcal{O}(\epsilon^2) \tag{2.1}$$

We can apply this linearization to neural nets. The source of non-linearity for neural nets lie in the elementwise non-linear activations (like ReLU, sigmoid) and pooling operators. It is easy to see that to linearize the entire neural network, one must only linearize such non-linearities.

2.3.1 Special case: ReLU and MaxPool

For the ReLU nonlinearity, the Taylor approximation is locally exact and simple to compute, as the derivative $\frac{\mathrm{d}\sigma(z)}{\mathrm{d}z}$ is either 0 or 1 (except at $z = 0$, where it is undefined). A similar statement holds for max-pooling. Going back to the definition in Equation 2.1, for piecewise linear nets there exist $\epsilon > 0$ such that the super-linear terms are zero, *i.e.*; $f(\mathbf{x} + \Delta\mathbf{x}) = f(\mathbf{x}) + \nabla_x f(\mathbf{x})^T(\Delta\mathbf{x})$ exactly.

2.3.2 What information does the gradient capture?

Consider piecewise linear functions with the non-differentiability at zero. Also consider neural networks with no external bias units among the model parameters. For such cases we see that the affine approximation reduces to a linear approximation, *i.e.*; the overall 'bias' term is zero. This means the following: let $\mathbf{x}_0$ be a point arbitrarily close to zero such that $f(\mathbf{x}_0 + \Delta\mathbf{x}) = f(\mathbf{x}_0) + \nabla_x f(\mathbf{x}_0)^T(\Delta\mathbf{x})$. Then, given $f(\mathbf{x}_0) \to 0$,

$\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0 \rightarrow \mathbf{x}$ and the fact that $\mathbf{x}$ and $\mathbf{x}_0$ have the same gradient, we have the following - $f(\mathbf{x}) = \nabla_x f(\mathbf{x})^T \mathbf{x}$. *In other words, the output can be obtained by multiplying the gradient with the input.* We find that this relation holds approximately even for ReLU nets with a bias unit as they are typically quite small (*e.g.* VGG19). Thus in this case the gradient captures almost all of the information required to make a decision. However, this breaks down for ReLU nets with batch normalization (*e.g.* ResNets) as they introduce their own mean subtraction terms. Even in such cases, one can always arbitrarily increase the scale of the input image such that the batch normalization terms are negligible and $f(\mathbf{x}) = \nabla_x f(\mathbf{x})^T \mathbf{x}$ holds in the limit.

As a corollary for other non-linearities, even with no external bias units, the overall 'bias' may be non-zero. For example, consider the hard-tanh nonlinearity, which is given by $\sigma(z) = z$ for $-1 \leq z \leq 1$, and saturates to -1 on the left and $+1$ on the right. In such a case, when the non-linearity saturates, the corresponding slope is zero and we get $\sigma(z + dz) = \sigma(z) = \pm 1$ bias depending on where it saturates. This reminds us that in general the gradient does not capture all information about the neural network, especially in the case of such saturating non-linearities.

2.3.3 Invariance to weight and architecture specification

One useful property of the Jacobian is that its dimensionality does not depend on the network architecture. For k output classes, and input dimension D , the Jacobian of a neural network is of dimension $D \times k$. This means that one can compare Jacobians of different architectures.

Another useful property is that for a given neural network architecture, different weight configurations can lead to the same Jacobian. One simple example of this is permutation symmetry of neurons in intermediate hidden layers. It is easy to see different permutations of neurons leave the Jacobian unchanged (as they have the same underlying function mapping). In general, because of redundancy of neural network models and non-convexity of the loss surface, several different weight configurations can end up having similar Jacobians.

Thus gradients and Jacobians naturally captures similarities between neural network mappings, making it desirable to use for knowledge transfer. Note that these properties hold trivially for output activations as well. Thus it seems sensible that both these quantities must be used for knowledge transfer. However, the important practical question remains: how exactly should this be done?

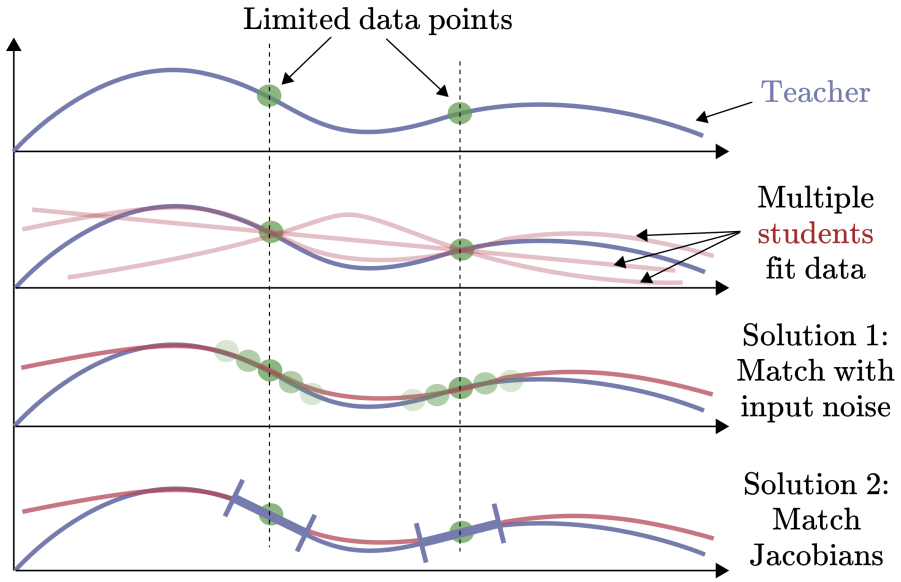


Figure 2.1 – Illustration of teacher-student learning in a simple 1D case. Here, x-axis is the input data, and y-axis denotes function outputs. Given a limited number of data points, there exist multiple student functions consistent with the data. How do we select the hypothesis closest to the teacher's? There are two equivalent solutions: either by augmenting the data set by adding noise to the inputs or by directly matching slopes (Jacobians) of the function at the data points.

2.4 Distillation

This problem of distillation is as follows: given a *teacher* network $\mathcal{T}$ which is trained on a dataset $\mathcal{D}$, we wish to enhance the training of a student network $\mathcal{S}$ on $\mathcal{D}$ using "hints" from $\mathcal{T}$. Classically, such "hints" involve activations of the output layer or some intermediate layers. Recent works (Czarnecki et al., 2017; Zagoruyko and Komodakis, 2017) sought to also match the Jacobians of $\mathcal{S}$ and $\mathcal{T}$. However, two aspects are not clear in these formalisms: (i) what penalty term must be used between Jacobians, and (ii) how this idea of matching Jacobians relates to simpler methods such as classical distillation or activation matching (Ba and Caruana, 2014; Hinton et al., 2015). To resolve these issues, we make the following claim.

Claim. *Matching Jacobians of two networks is equivalent to matching soft targets with noise added to the inputs during training.*

More concretely, we make the following proposition.

Proposition 1. *Consider the squared error cost function for matching soft targets of two neural networks with k-length targets ($\in \mathbb{R}^k$), given by $\ell(\mathcal{T}(\mathbf{x}), \mathcal{S}(\mathbf{x})) = \sum_{i=1}^{k}(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}))^2$, where $\mathbf{x} \in \mathbb{R}^D$ is an input data point. Let $\boldsymbol{\xi}$ $(\in \mathbb{R}^D) = \sigma$ $\mathbf{z}$ be a scaled version of a unit normal random variable $\mathbf{z}$ $\in \mathbb{R}^D$ with scaling factor $\sigma \in \mathbb{R}$. Then,*

$$\mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i=1}^{k}\Big(\mathcal{T}^i(\mathbf{x}+\boldsymbol{\xi})-\mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\Big)^2\right]$$
$$=\sum_{i=1}^{k}\Big(\mathcal{T}^i(\mathbf{x})-\mathcal{S}^i(\mathbf{x})\Big)^2$$
$$+\sigma^2\sum_{i=1}^{k}\|\nabla_x\mathcal{T}^i(\mathbf{x})-\nabla_x\mathcal{S}^i(\mathbf{x})\|_2^2+\mathcal{O}(\sigma^4).$$

Notice that in this expression, we have decomposed the loss function into two components: one representing the usual distillation loss on the samples, and the second regularizer term representing the Jacobian matching loss. The higher order error terms are small for small σ and can be ignored. The above proposition is a simple consequence of using the first-order Taylor series expansion around x. Note that the error term is exactly zero for piecewise-linear nets. An analogous statement is true for the case of cross entropy error between soft targets, leading to:

$$\mathbb{E}_{\boldsymbol{\xi}}\left[-\sum_{i=1}^{k}\mathcal{T}_s^i(\mathbf{x}+\boldsymbol{\xi})\log\Big(\mathcal{S}_s^i(\mathbf{x}+\boldsymbol{\xi})\Big)\right] \tag{2.2}$$
$$\approx-\sum_{i=1}^{k}\mathcal{T}_s^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x}))\;-\;\sigma^2\sum_{i=1}^{k}\frac{\nabla_x\mathcal{T}_s^i(\mathbf{x})^T\nabla_x\mathcal{S}_s^i(\mathbf{x})}{\mathcal{S}_s^i(\mathbf{x})}$$

where $\mathcal{T}_s^i(\mathbf{x})$ denotes the same network $\mathcal{T}^i(\mathbf{x})$ but with a softmax or sigmoid (with temperature parameter T if needed) added at the end. We do not write the super-linear error terms for convenience. This shows that the Jacobian matching loss does not need to be specified seperately, and that it arises naturally from the choice of activation matching loss and the noise model. This observation can be used in practice to pick appropriate loss function by choosing a specific noise model of interest.

These statements show that matching Jacobians is a natural consequence of matching not only the raw network outputs at given data points, but also at the infinitely many data points nearby. This is illustrated in Figure 2.1, which shows that by matching on a noise-augmented dataset, the student is able to mimic the teacher better.

We can use the idea of noise augmentation to derive regularizers for the case of regular neural network training as well. These regularizers seek to make the underlying model *robust* to small amounts of noise added to the inputs.

Proposition 2. *Consider the squared error cost function for training a neural network with k targets, given by $\ell(y(\mathbf{x}),\mathcal{S}(\mathbf{x}))=\sum_{i=1}^{k}(y^i(\mathbf{x})-\mathcal{S}^i(\mathbf{x}))^2$, where $\mathbf{x}\in\mathbb{R}^D$ is an input*

data point, and $y^i(\mathbf{x})$ *is the* i^{th} *target output. Let* $\boldsymbol{\xi}$ $(\in \mathbb{R}^D) = \sigma$ $\mathbf{z}$ *be a scaled version of a unit normal random variable* $\mathbf{z}$ $\in \mathbb{R}^D$ *with scaling factor* $\sigma \in \mathbb{R}$*. Then the following is true.*

$$\mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i=1}^{k}\Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\Big)^2\right]$$
$$= \sum_{i=1}^{k}\Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})\Big)^2 + \sigma^2\sum_{i=1}^{k}\|\nabla_x\mathcal{S}^i(\mathbf{x})\|_2^2 + \mathcal{O}(\sigma^4)$$

A statement similar to Proposition 2 has been previously derived by Bishop (1995), who observed that the regularizer term for linear models corresponds exactly to the well-known Tikhonov regularizer. This regularizer was also proposed by Drucker and Le Cun (1992). The ℓ_2 weight decay regularizer for neural networks can be derived by applying this regularizer layer-wise separately. However, we see here that a more appropriate way to ensure noise robustness is to penalize the norm of the gradient rather than weights. We can derive a similar result for the case of cross-entropy error as well, which is given by -

$$\mathbb{E}_{\boldsymbol{\xi}}\left[-\sum_{i=1}^{k} y^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x}+\boldsymbol{\xi}))\right] \tag{2.3}$$
$$\approx -\sum_{i=1}^{k} y^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x})) \;+\; \sigma^2\sum_{i=1}^{k} y^i(\mathbf{x})\frac{\|\nabla_x\mathcal{S}_s^i(\mathbf{x})\|_2^2}{\mathcal{S}_s^i(\mathbf{x})^2}$$

We notice here again that the regularizer involves $\mathcal{S}_s^i(\mathbf{x})$, which has the sigmoid / softmax nonlinearity applied on top of the final layer of $\mathcal{S}^i(\mathbf{x})$. Deriving all the above results is a simple matter of using first-order Taylor series expansions, and additionally a second-order expansion for log in the case of Equation 2.3. Proof is provided in the supplementary material.

Note that we can re-write the penalties for cross entropy error in a more numerically stable form. In general, we found that the penalties for squared error worked better experimentally and were easier to tune. As a result, we use squared error loss for distillation.

Why does Jacobian matching improve performance? One reason is that Jacobian matching is derived from the *expected value* of the activation matching loss with noise, and computing this expected loss is intractable in practice. However it can be approximated by averaging over a large number N of noise instances, *i.e.* a Monte Carlo approximation. This is a form of data augmentation with noise. Thus with Jacobian matching we

analytically perform an otherwise intractable data augmentation procedure. For example, for CIFAR100 we found that we needed $N \sim 10^4$ which is intractable in practice. A more learning-theoretic reason for improvement is presented in the next section, mainly applied to transfer learning.

However it can be approximated by averaging over a large number N of noise instances, *i.e.* a Monte Carlo approximation. This scheme increases the computational cost by a factor of N. In contrast, Jacobian matching only increases computational cost by a factor of 2, because of the double backward pass.

2.4.1 Approximating the Full Jacobian

One can see that both in the case of Proposition 1 and 2, we are required to compute the full Jacobian. This is computationally expensive, and sometimes unnecessary. For example, Equation 2.3 requires only the terms where $y^i(\mathbf{x})$ is non-zero.

In general, we can approximate the summation of Jacobian terms with the one with largest magnitude. However, we cannot estimate this without computing the Jacobians themselves. As a result, we use a heuristic where the only output variable involving the correct answer $c \in [1, k]$ is used for computing the Jacobian. This corresponds to the case of Equation 2.3. Alternately, if we do not want to use the labels, we may instead use the output variable with the largest magnitude, as it often corresponds to the right label (for good models).

2.5 Transfer Learning

We now apply our Jacobian matching machinery to transfer learning problems. In computer vision, transfer learning is often done by fine-tuning (Yosinski et al., 2014), where models pre-trained on a large *source* dataset $\mathcal{D}_s$, such as Imagenet (Russakovsky et al., 2015), are used as initialization for training on another smaller *target* dataset $\mathcal{D}_t$. Practically, this means that the architecture used for fine-tuning must be the same as that of the pre-trained network, which is restrictive. We would like to develop transfer learning methods where the architectures of the pre-trained network and target "fine-tuned" network can be arbitrarily different.

One way to achieve this is by distillation: we can match output activations of a pre-trained teacher network and an untrained student network. However, this procedure is not general as the target dataset may not share the same label space as the source dataset. To overcome this, we can design the student network to have two sets of outputs (or two output "branches"), one with the label space of the smaller target dataset, while the other with that of the larger source dataset. This leads to the method proposed by Li and Hoiem (2016), called "Learning without Forgetting" (**LwF**). Note that similar

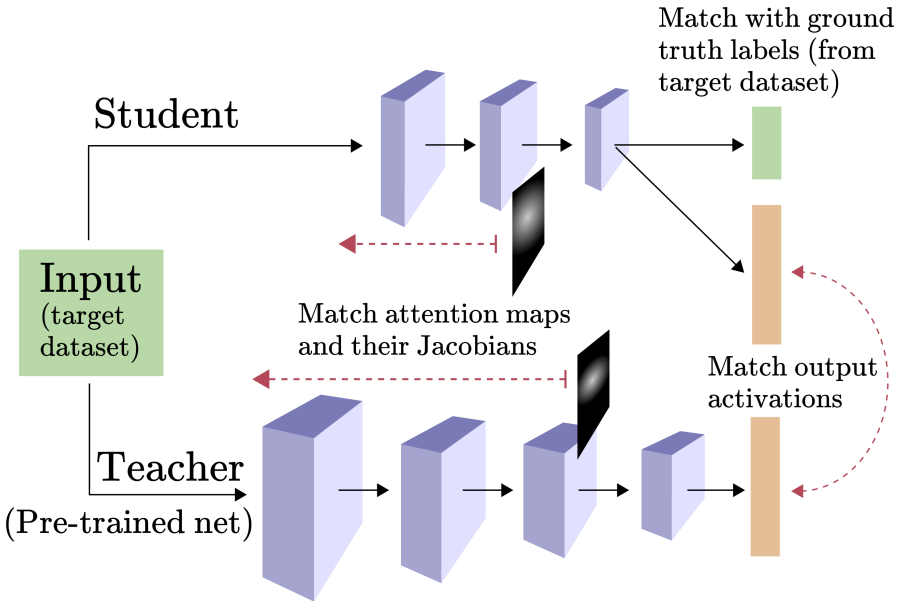


Figure 2.2 – Illustration of our proposed method for transfer learning. We match the output activations of a pre-trained Imagenet network similar to LwF (Li and Hoiem, 2016). We also match aggregated activations or "attention" maps between networks, similar to the work of Zagoruyko and Komodakis (2017). We propose to match Jacobians of (aggregated) attention maps w.r.t. inputs.

methods were concurrently developed by Jung et al. (2016) and Furlanello et al. (2016). In this method, the student network is trained with a composite loss function involving two terms, one in each output branch. The two objectives are **(1)** matching ground truth labels on the target dataset, and **(2)** matching the activations of the student network and a pre-trained teacher network on the target dataset. This is illustrated in Figure 2.2. Crucially, these losses are matched only on the target dataset, and the source data is untouched. This is conceptually different from distillation, where the teacher network is trained on the dataset being distilled. In LwF, the pre-trained teacher is not trained on the target dataset.

This makes it problematic to apply our Jacobian matching framework to LwF. For distillation, it is clear that adding input noise (or Jacobian matching) can improve overall matching as shown in Figure 2.1. For the case of LwF, it is not clear whether improving matching between teacher and student will necessarily improve transfer learning performance. This is especially because the teacher is not trained on the target dataset, and can potentially produce noisy or incorrect results on this unseen data. To resolve this ambiguity, we shall now connect LwF with distillation.

2.5.1 LwF as Distillation

In the discussion below we shall only consider the distillation-like loss of LwF, and ignore the branch which matches ground truth labels. For LwF to work well, it must be the case that the activations of the pre-trained teacher network on the target dataset must contain information about the source dataset (*i.e.*; Imagenet). The attractiveness of LwF lies in the fact that this is done without explicitly using Imagenet. Here, we make the claim that *LwF approximates distillation on (a part of) Imagenet.*

Let $f(\cdot)$ be an untrained neural network, $g(\cdot)$ be a pre-trained network, $\mathbf{x}, \mathbf{y}$ be the input image and corresponding ground truth label respectively. Let $|\mathcal{D}|$ be the size of the dataset $\mathcal{D}$. Let us denote $\rho(\mathbf{x}) = \ell(f(\mathbf{x}), g(\mathbf{x}))$ for convenience, where $\ell(\cdot, \cdot)$ is a loss function. Assume Lipschitz continuity for $\rho(\mathbf{x})$ with Lipschitz constant K, and distance metric $\psi_{\mathbf{x}}$ in the input space

$$\|\rho(\mathbf{x}_1) - \rho(\mathbf{x}_2)\| \leq \mathrm{K}\psi_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2) \tag{2.4}$$

Note here that the distance in the input space need not be in terms of pixelwise distances, but can also be a learnt feature distance, for example. Let us also define an assymetric version of the Hausdorff distance between two sets A, B:

$$\mathcal{H}_a(A, B) = \sup_{a \in A} \inf_{b \in B} \psi_{\mathbf{x}}(a, b). \tag{2.5}$$

Note that this is no longer a valid distance metric unlike the Hausdorff. Given these assumptions, we are now ready to state our result.

Proposition 3. *Given the assumptions and notations described above, we have*

$$\frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \sim \mathcal{D}_s} \ell(f(\mathbf{x}), g(\mathbf{x})) \leq \max_{\mathbf{x} \sim \mathcal{D}_t} \ell(f(\mathbf{x}), g(\mathbf{x})) + \mathrm{K}\mathcal{H}_a(\mathcal{D}_s, \mathcal{D}_t) \tag{2.6}$$

On the left side of 2.6 we have the distillation loss on the source dataset, and on the right we have a max-loss term on the target dataset. Note that the LwF loss is an average loss on the target dataset. As expected, the slack terms in the inequality depends on the distance between the source and target datasets (2.6). This bounds a loss related to the LwF loss (*i.e.* max-loss instead of average) with the distillation loss. If the Hausdorff distance is small, then reducing the max-loss would reduce the distillation loss as well. A similar theory was previously presented by Ben-David et al. (2010), but with different formalisms. Our formalism allows us to connect with Jacobian matching, which is our primary objective. Note that this inequality can also be viewed as a learning-theoretic generalization bound for distillation by replacing the source and target datasets with train and test sets for distillation instead.

In practice, the target dataset is often much smaller than Imagenet and has different overall statistics. For example, the target dataset could be a restricted dataset of flower images. In such a case, we can restrict the source dataset to its "best" subset, in particular with all the irrelevant samples (those far from target dataset) removed. This would make the Hausdorff distance smaller, and provide a tighter bound. In our example, this involves keeping only flowers from Imagenet.

This makes intuitive sense as well: if the source and target datasets are completely different, we do not expect transfer learning (and thus LwF) to help. The greater the overlap between source and target datasets, the smaller is the Hausdorff distance, the tighter is the bound, and the more we expect knowledge transfer to help. Our results capture this intuition in a rigorous manner. In addition, this predicts that Lipschitz-smooth teacher neural nets that produce small feature distance between source and target images are expected to do well in transfer learning. Lipschitz-smoothness of models has been previously related to adversarial noise robustness (Cisse et al., 2017), and to learning theory as a sufficient condition for generalization (Xu and Mannor, 2012). It is interesting that this relates to transfer learning as well.

More importantly, this establishes LwF as an approximate distillation method. The following result motivates input noise added to the target dataset.

Corollary. *For any superset* $\mathrm{D'}_t \supseteq \mathcal{D}_t$ *of the target dataset,* $\mathcal{H}_a(\mathcal{D}_s, \mathcal{D'}_t) \leq \mathcal{H}_a(\mathcal{D}_s, \mathcal{D}_t)$

Thus if we augment the target dataset $\mathcal{D}_t$ by adding noise, we expect the bound in Proposition 3 to get tighter. This is because when we add noise to points in $\mathcal{D}_t$, the minimum distance between points from $\mathcal{D}_s$ to $\mathcal{D}_t$ decreases. Proofs are provided in the supplementary material.

To summarize, we have showed that a loss related to the LwF loss (max-loss) is an upper bound on the true distillation loss. Thus by minimizing the upper bound, we can expect to reduce the distillation loss also.

Now that input noise and thus Jacobian matching is well motivated, we can incorporate these losses into LwF. When we implemented this for practical deep networks we found that the optimizer was not able to reduce the Jacobian loss at all. We conjecture that it might be because of a vanishing gradient effect / network degeneracy on propagation of second order gradients through the network (and not the first). Hence we need an alternate way to match Jacobians.

2.5.2 Matching attention maps

It is often insufficient to match only output activations between a teacher and a student network, especially when both networks are deep. In such cases we can consider

matching intermediate feature maps as well. In general it is not possible to match the full feature maps between an arbitrary teacher and student network as they may have different architectures, and features sizes may never match at any layer. However, for modern convolutional architectures, spatial sizes of certain features often match across architectures even when the number of channels does not. Zagoruyko and Komodakis (2017) noticed that it in such cases it helps to match channelwise aggregated activations, which they call *attention* maps. Specifically, this aggregation is performed by summing over squared absolute value of channels of a feature activation Z, and is given by -

$$A = AttMap(Z) = \sum_{i \in channels} |Z_i|^2 \tag{2.7}$$

Further, the loss function used to match these activations is

$$\text{Match Activations} = \left|\left| \frac{A_t}{\|A_t\|_2} - \frac{A_s}{\|A_s\|_2} \right|\right|_2 \tag{2.8}$$

Here, A_t, A_s are the attention maps of the teacher and student respectively. Zagoruyko and Komodakis (2017) note that this choice of loss function is especially crucial.

Incorporating Jacobian loss

As the activation maps have large spatial dimensions, it is computationally costly to compute the full Jacobians. We hence resort to computing approximate Jacobians in the same manner as previously discussed. In this case, this leads to picking the pixel in the attention map with the largest magnitude, and computing the Jacobian of this quantity w.r.t. input. We compute the index (i, j) of this maximum-valued pixel for the teacher network and use the same index to compute the student's Jacobian. We then use a loss function similar to Equation 2.8, given by

$$\text{Match gradients} = \left|\left| \frac{\nabla_x f(\mathbf{x})}{\|\nabla_x f(\mathbf{x})\|_2} - \frac{\nabla_x g(\mathbf{x})}{\|\nabla_x g(\mathbf{x})\|_2} \right|\right|_2^2 \tag{2.9}$$

Justification for Jacobian loss

We can show that the above loss term corresponds to adding a noise term $\boldsymbol{\xi}_f \propto \|\nabla_x f(\mathbf{x})\|_2^{-1}$ for $f(\mathbf{x})$ and $\boldsymbol{\xi}_g \propto \|\nabla_x g(\mathbf{x})\|_2^{-1}$ for $g(\mathbf{x})$ for the distillation loss. From the first order Taylor series expansion, we see that $g(x + \boldsymbol{\xi}) = g(x) + \boldsymbol{\xi}_g \nabla_x g(\mathbf{x})$. Thus for networks $f(\cdot)$ and $g(\cdot)$ with different Jacobian magnitudes, we expect different responses for the same noisy inputs. Specifically, we see that $\mathbb{E}_{\boldsymbol{\xi}_g} \|g(x + \boldsymbol{\xi}_g) - g(x)\|_2^2 = \sigma_g^2 \|\nabla_x g(\mathbf{x})\|_2^2 = \sigma^2 \frac{\|\nabla_x g(\mathbf{x})\|_2^2}{\|\nabla_x g(\mathbf{x})\|_2^2} = \sigma^2$ for a gaussian model with covariance matrix being σ times the identity.

2.6 Experiments

We perform three experiments to show the effectiveness of using Jacobians. First, we perform distillation in a limited data setting on the CIFAR100 dataset (Krizhevsky and Hinton, 2009). Second, we show on that same dataset that penalizing gradient norm increases stability of networks to random noise. Finally, we perform transfer learning experiments on the MIT Scenes dataset (Quattoni and Torralba, 2009). We provide more detail about the experimental setups in the supplementary material.

2.6.1 Distillation

For the distillation experiments, we use VGG-like (Simonyan and Zisserman, 2014) architectures with batch normalization. The main difference is that we retain the convolutional layers and have one fully connected layer rather than three. Our workflow is as follows. First, a 9-layer "teacher" network is trained on the full CIFAR100 dataset. Then, a smaller 4-layer "student" network is trained, but this time on small subsets rather than the full dataset. As the teacher is trained on much more data than the student, we expect distillation to improve the student's performance.

A practical scenario where this would be useful is the case of architecture search and ensemble training, where we require to train many candidate neural network architectures on the same task. Distillation methods can help speed up such methods by using already trained networks to accelerate training of newer models. One way to achieve acceleration is by using less data to train the student.

We compare our methods against the following baselines.

1. **Cross-Entropy (CE) training** – Here we train the student using only the ground truth (hard labels) available with the dataset without invoking the teacher network.
2. **CE + match activations** – This is the classical form of distillation, where the activations of the teacher network are matched with that of the student. This is weighted with the cross-entropy term which uses ground truth targets.
3. **Match activations only** – Same as above, except that the cross-entropy term is not used in the loss function.

We compare these methods by appending the Jacobian matching term in each case. In all cases, we use the squared-error distillation loss shown in Proposition 1. We found that squared loss worked much better than the cross-entropy loss for distillation and it was much easier to tune.

From Table 2.1 we can conclude that (1) it is generally beneficial to do any form of

Table 2.1 – Distillation performance on the CIFAR100 dataset. Table shows test accuracy (%). We find that matching both activations and Jacobians along with cross-entropy error performs the best for limited-data settings. The student network is VGG-4 while the teacher is a VGG-9 network which achieves 64.78% accuracy.

# of Data points per class →	1	5	10	50	100	500
Cross-Entropy (CE)	5.69	13.9	20.03	37.6	44.92	54.28
CE + match activations (A)	12.13	26.97	33.92	46.47	50.92	**56.65**
CE + match Jacobians (J)	6.78	23.94	32.03	45.71	51.47	53.44
CE + match {A + J}	**13.78**	**33.39**	**39.55**	**49.49**	**52.43**	54.57
Match activations (A) only	10.73	28.56	33.6	45.73	50.15	56.59
Match {A + J}	13.09	33.31	38.16	47.79	50.06	51.33

distillation to improve performance, (2) matching Jacobians along with activations outperforms matching only activations in low-data settings, (3) not matching Jacobians is often beneficial for large data.

One interesting phenomenon we observe is that having a cross-entropy (CE) error term is often not crucial to achieve good performance. It performs only slightly worse than using ground truth labels.

For Table 2.1, we see that when training with activations, Jacobians and regular cross-entropy training (fourth row), we reach an accuracy of 52.43% when training with 100 data points per class. We observe that the overall accuracy of raw training using the full dataset is 54.28%. Thus we are able to reach close to the full training accuracy using only about $1/5^{th}$ of the data.

2.6.2 Noise robustness

We perform experiments where we penalize the gradient norm to improve robustness of models to random noise. We train 9-layer VGG networks on CIFAR100 with varying amount of the regularization strength (λ), and measure their classification accuracy in presence of noise added to the normalized images. From Table 2.2 we find that using higher regularization strengths is better for robustness, even when the initial accuracy at the zero-noise case is lower. This aligns remarkably well with the theory. Surprisingly, we find that popular regularizers such as ℓ_2 regularization and dropout (Srivastava et al., 2014) are not robust.

Table 2.2 – Robustness of various VGG-9 models to gaussian noise added to CIFAR100 images at test time. Table shows test accuracy (%). λ is the regularization strength of the gradient-norm penalty regularizer. $\gamma = 1e-3$ is the ℓ_2 regularization strength and $p = 0.25$ is the dropout value. We see that the gradient-norm penalty can be remarkably robust to noise, unlike ℓ_2 regularization and dropout.

Noise (σ) $\rightarrow$	0	0.1	0.2	0.3	0.4
$\lambda = 0$	64.78	61.9 ± 0.07	47.53 ± 0.23	29.63 ± 0.16	17.69 ± 0.17
$\lambda = 0.01$	64.67	61.85 ± 0.15	49.47 ± 0.07	32.24 ± 0.28	19.63 ± 0.17
$\lambda = 0.1$	65.62	$\mathbf{63.36 \pm 0.18}$	53.57 ± 0.23	37.38 ± 0.18	23.99 ± 0.19
$\lambda = 1.0$	63.59	62.66 ± 0.13	57.53 ± 0.17	47.48 ± 0.14	35.43 ± 0.11
$\lambda = 10.0$	61.37	60.78 ± 0.05	$\mathbf{58.28 \pm 0.13}$	$\mathbf{52.82 \pm 0.10}$	$\mathbf{44.96 \pm 0.19}$
ℓ_2 reg.	**66.92**	60.41 ± 0.27	39.93 ± 0.28	23.32 ± 0.25	13.76 ± 0.16
Dropout	66.8	61.53 ± 0.10	44.34 ± 0.19	26.70 ± 0.24	15.77 ± 0.11

2.6.3 Transfer Learning

For transfer learning, our objective is to improve training on the target dataset (MIT Scenes) by using Imagenet pre-trained models. Crucially, we want our MIT Scenes model to have a different architecture than the Imagenet model. The teacher model we use is a ResNet-34 (He et al., 2016) pre-trained on Imagenet, while the student model is an untrained VGG-9 model with batch normalization. We choose VGG-9 because its architecture is based on fundamentally different design principles than ResNets. In principle we can use any architecture for the student. We modify this VGG-9 architecture such that it has two sets of outputs, one sharing the label space with Imagenet (1000 classes), and another with MIT Scenes (67 classes). The pre-final layer is common to both outputs.

Our baselines are the following.

1. **Cross-Entropy (CE) training of student with ground truth** – Here we ignore the VGG-9 branch with 1000 classes and optimize the cross-entropy error on MIT Scenes data. The loss function on this branch is always the same for all methods.

2. **CE on pre-trained network** – This is exactly the same as the first baseline, except that the VGG-9 model is initialized from Imagenet pre-trained weights. We consider this as our "oracle" method and strive to match its performance.

3. **CE + match activations (LwF)** – This corresponds to the method of Li and Hoiem (2016), where the Imagenet branch output activations of the student are matched with those of the teacher.

4. **CE + match {activations + attention}** – This corresponds to the method of Zagoruyko and Komodakis (2017), where attention maps are matched between some intermediate layers.

We add our Jacobian matching terms to the baselines 3 and 4. We provide our results in Table 2.3. In all cases, we vary the number of images per class on MIT Scenes to observe the performance on low-data settings as well. In this case we average our results over two runs by choosing different random subsets.

Table 2.3 – Transfer Learning from Imagenet to MIT Scenes dataset. Table shows test accuracy (%). The student network (VGG9) is trained from scratch unless otherwise mentioned. The teacher network used is a pre-trained ResNet34. Here CE denote training with cross-entropy loss, J denotes Jacobian and att denotes attention. Results are averaged over two runs.

# data per class →	5	10	25	50	Full
CE on untrained student	11.64	20.30	35.19	46.38	59.33
CE on pre-trained student (Oracle)	**25.93**	**43.81**	**57.65**	**64.18**	**71.42**
CE + match activations (A)	17.08	27.13	45.08	55.22	65.22
CE + match {A + J}	17.88	28.25	45.26	56.49	66.04
CE + match {A + att.}	16.53	28.35	46.01	57.80	**67.24**
CE + match {A + att. + J}	**18.02**	**29.25**	**47.31**	**58.35**	**67.31**

Table 2.4 – Ablation experiments over choice of feature matching depth. ($\mathcal{T}$, $\mathcal{S}$) denotes teacher (ResNet34) and student (VGG9) feature depths. These pairs are chosen such that resulting features have same spatial dimensions. We see that matching the shallowest feature works the best. Results are averaged over two runs.

Feature matching depth ($\mathcal{T}$, $\mathcal{S}$)	(7, 2)	(15, 4)	(27, 6)	(33, 8)
Accuracy (%)	**22.39**	21.98	20.45	20.03
gradient loss reduction (%)	**25.88**	15.59	11.55	1.25

Experiments show that matching activations and attention maps increases performance at all levels of data size. It also shows that Jacobians improve performance of all these methods. However, we observe that none of the methods match the oracle performance of using a pre-trained model. The gap in performance is especially large at intermediate data sizes of 10 and 25 images per class.

Table 2.5 – Ablation experiments over the computation of gradient. Here, s is the size of the attention map. "Full" is global average pooling, and "None" is no average pooling. We see that using average pooling while computing Jacobians helps performance. Results are averaged over two runs.

Average Pool Window size	Full	$s/3$	$s/5$	$s/7$	None
Accuracy (%)	20.00	21.20	**21.87**	21.09	19.74

Which features to match?

An important practical question is the choice of intermediate features to compute attention maps for matching. The recipe followed by Zagoruyko and Komodakis (2017) for ResNets is to consider features at the end of a residual block. [1] As there are typically 3-5 max-pooling layers in most modern networks, we have 3-5 intermediate features to match between any typical teacher and student network. Note that we require the attention maps (channelwise aggregated features) to be of similar spatial size to match. Zagoruyko and Komodakis (2017) match at all such possible locations.

However, computing Jacobians at all such locations is computationally demanding and perhaps unnecessary. We observe that if we compute Jacobians at later layers, we are still not able to reduce training Jacobian loss, possibly due to a "second-order" vanishing gradient effect. At suitable intermediate layers, we see that loss reduction occurs. This is reflected in Table 2.4, where we vary the feature matching depth and observe performance. We observe that the Jacobian loss reduction (during training) is highest for the shallowest layers, and this corresponds to good test performance as well. These ablation experiments are done on the MIT Scenes dataset picking only 10 points per class.

Feature Pooling to compute Jacobians

Instead of considering a single pixel per attention map to compute Jacobians, we can aggregate a large number of large-magnitude pixels. One way to do this is by average pooling over the attention map, and then picking the maximum pixel over the average pooled map. In Table 2.5 we vary the window size for average pooling and observe performance. We observe that it is beneficial to do average pooling, we find that using a window size of (feature size)/5 works the best. These ablation experiments are done on the MIT Scenes dataset picking only 10 points per class.

[1] A residual block is the set of all layers in between two consecutive max-pooling layers in a ResNet. All features in a block have the same dimensions.

2.7 Conclusion

In this chapter we considered matching Jacobians of deep neural networks for knowledge transfer. Viewing Jacobian matching as a form of data augmentation with gaussian noise motivates their usage, and also informs us about the loss functions to use. We also connected a recent transfer learning method (LwF) to distillation, enabling us to incorporate Jacobian matching.

Despite our advances, there is still a large gap between distillation-based methods and the oracle method of using pre-trained nets for transfer learning. Future work can focus on closing this gap by considering more structured forms of data augmentation than simple noise addition.

Appendix

2.8 Proof of Proposition 1

Proposition. *Consider the* ***squared error*** *cost function for matching soft targets of two neural networks with k-length targets ($\in \mathbb{R}^k$), given by $\ell(\mathcal{T}(\mathbf{x}), \mathcal{S}(\mathbf{x})) = \sum_{i=1}^{k}(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}))^2$, where $\mathbf{x} \in \mathbb{R}^D$ is an input data point. Let $\boldsymbol{\xi}$ $(\in \mathbb{R}^D) = \sigma$ $\mathbf{z}$ be a scaled version of a unit normal random variable $\mathbf{z}$ $\in \mathbb{R}^D$ with scaling factor $\sigma \in \mathbb{R}$. Then,*

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big(\mathcal{T}^i(\mathbf{x}+\boldsymbol{\xi}) - \mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\Big)^2 \\
&= \sum_{i=1}^{k} \Big(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})\Big)^2 + \sigma^2 \sum_{i=1}^{k} \|\nabla_x \mathcal{T}^i(\mathbf{x}) - \nabla_x \mathcal{S}^i(\mathbf{x})\|_2^2 + \mathcal{O}(\sigma^4)
\end{aligned}$$

Proof. There exists σ and $\boldsymbol{\xi}$ small enough that first-order Taylor series expansion holds true

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big(\mathcal{T}^i(\mathbf{x}+\boldsymbol{\xi}) - \mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\Big)^2 \\
=&\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big(\mathcal{T}^i(\mathbf{x}) + \boldsymbol{\xi}^\top \nabla_x \mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}) - \boldsymbol{\xi}^\top \nabla_x \mathcal{S}^i(\mathbf{x})\Big)^2 && \text{(Taylor series)} \\
&+\mathcal{O}(\sigma^4) \\
=&\sum_{i=1}^{k} \Big(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})\Big)^2 && \text{(expand the square)} \\
&+\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big[\boldsymbol{\xi}^\top \Big(\nabla_x \mathcal{T}^i(\mathbf{x}) - \nabla_x \mathcal{S}^i(\mathbf{x})\Big)\Big]^2 \\
&+2\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big[\boldsymbol{\xi}^\top \Big(\nabla_x \mathcal{T}^i(\mathbf{x}) - \nabla_x \mathcal{S}^i(\mathbf{x})\Big)\Big] [\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})] && \text{(=0, as } \boldsymbol{\xi} \text{ is zero mean)}
\end{aligned}$$

$$
\begin{aligned}
&+\mathcal{O}(\sigma^4) \\
&=\sum_{i=1}^{k}\Big(\mathcal{T}^i(\mathbf{x})-\mathcal{S}^i(\mathbf{x})\Big)^2 \\
&+\sum_{i=1}^{k}\Big[\Big(\nabla_x\mathcal{T}^i(\mathbf{x})-\nabla_x\mathcal{S}^i(\mathbf{x})\Big)^\top \mathbb{E}_{\boldsymbol{\xi}}\boldsymbol{\xi}\boldsymbol{\xi}^\top\Big(\nabla_x\mathcal{T}^i(\mathbf{x})-\nabla_x\mathcal{S}^i(\mathbf{x})\Big)\Big] \quad \text{(linearity of expectation)} \\
&+\mathcal{O}(\sigma^4) \\
&=\sum_{i=1}^{k}\Big(\mathcal{T}^i(\mathbf{x})-\mathcal{S}^i(\mathbf{x})\Big)^2 \\
&+\sum_{i=1}^{k}\sigma^2\|\nabla_x\mathcal{T}^i(\mathbf{x})-\nabla_x\mathcal{S}^i(\mathbf{x})\|^2 \qquad (\mathbb{E}_{\boldsymbol{\xi}}\boldsymbol{\xi}\boldsymbol{\xi}^\top=\sigma I) \\
&+\mathcal{O}(\sigma^4)
\end{aligned}
$$

□

Proposition. *Consider the **cross-entropy** cost function for matching soft targets of two neural networks with k-length targets ($\in \mathbb{R}^k$), given by $\ell(\mathcal{T}(\mathbf{x}),\mathcal{S}(\mathbf{x}))=\sum_{i=1}^{k}\mathcal{T}^i(\mathbf{x})\log\left(\mathcal{S}^i(\mathbf{x})\right)$, where $\mathbf{x}\in\mathbb{R}^D$ is an input data point. Let $\boldsymbol{\xi}$ $(\in\mathbb{R}^D)=\sigma$ $\mathbf{z}$ be a scaled version of a unit normal random variable $\mathbf{z}$ $\in\mathbb{R}^D$ with scaling factor $\sigma\in\mathbb{R}$. Then,*

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}}-\sum_{i=1}^{k}\mathcal{T}_s^i(\mathbf{x}+\boldsymbol{\xi})\log\Big(\mathcal{S}_s^i(\mathbf{x}+\boldsymbol{\xi})\Big) \\
&=-\sum_{i=1}^{k}\mathcal{T}_s^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x}))\;-\;\sigma^2\sum_{i=1}^{k}\frac{\nabla_x\mathcal{T}_s^i(\mathbf{x})^T\nabla_x\mathcal{S}_s^i(\mathbf{x})}{\mathcal{S}_s^i(\mathbf{x})}-\mathcal{O}(\sigma^4)
\end{aligned}
$$

Proof. There exists σ and $\boldsymbol{\xi}$ small enough that first-order Taylor series expansion holds true

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}}-\sum_{i=1}^{k}\mathcal{T}^i(\mathbf{x}+\boldsymbol{\xi})\log\Big(\mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\Big) \\
&=\mathbb{E}_{\boldsymbol{\xi}}-\sum_{i=1}^{k}\Big(\mathcal{T}^i(\mathbf{x})+\boldsymbol{\xi}^\top\nabla_{\mathbf{x}}\mathcal{T}^i(\mathbf{x})\Big)\Big(\log\mathcal{S}^i(\mathbf{x})+\boldsymbol{\xi}^\top\nabla_{\mathbf{x}}\log\mathcal{S}^i(\mathbf{x})\Big) \quad \text{(Taylor series)} \\
&-\mathcal{O}(\sigma^4) \\
&=-\sum_{i=1}^{k}\mathcal{T}^i(\mathbf{x})\log\mathcal{S}^i(\mathbf{x})
\end{aligned}
$$

$$
\begin{aligned}
&-\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \mathcal{T}^i(\mathbf{x}) \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) \\
&-\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} (\mathcal{T}^i(\mathbf{x}) \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}))(\log \mathcal{S}^i(\mathbf{x}) \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \mathcal{T}^i(\mathbf{x})) && (=0, \text{ as } \boldsymbol{\xi} \text{ is zero mean}) \\
&-\mathcal{O}(\sigma^4) \\
&= -\sum_{i=1}^{k} \mathcal{T}^i(\mathbf{x}) \log \mathcal{S}^i(\mathbf{x}) \\
&-\sum_{i=1}^{k} \nabla_{\mathbf{x}} \mathcal{T}^i(\mathbf{x})^\top \mathbb{E}_{\boldsymbol{\xi}} \boldsymbol{\xi} \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) && (\text{linearity of expectation}) \\
&-\mathcal{O}(\sigma^4) \\
&= -\sum_{i=1}^{k} \mathcal{T}^i(\mathbf{x}) \log \mathcal{S}^i(\mathbf{x}) \\
&-\sigma^2 \sum_{i=1}^{k} \nabla_{\mathbf{x}} \mathcal{T}^i(\mathbf{x})^\top \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) && (\mathbb{E}_{\boldsymbol{\xi}} \boldsymbol{\xi} \boldsymbol{\xi}^\top = \sigma I) \\
&-\mathcal{O}(\sigma^4)
\end{aligned}
$$

□

2.9 Proof of Proposition 2

Proposition. *Consider the* ***squared error*** *cost function for training a neural network with k targets, given by $\ell(y(\mathbf{x}), \mathcal{S}(\mathbf{x})) = \sum_{i=1}^{k} (y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}))^2$, where $\mathbf{x} \in \mathbb{R}^D$ is an input data point, and $y^i(\mathbf{x})$ is the i^{th} target output. Let $\boldsymbol{\xi}$ $(\in \mathbb{R}^D) = \sigma$ $\mathbf{z}$ be a scaled version of a unit normal random variable $\mathbf{z}$ $\in \mathbb{R}^D$ with scaling factor $\sigma \in \mathbb{R}$. Then the following is true.*

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x} + \boldsymbol{\xi}) \Big)^2 \\
&= \sum_{i=1}^{k} \Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}) \Big)^2 + \sigma^2 \sum_{i=1}^{k} \| \nabla_x \mathcal{S}^i(\mathbf{x}) \|_2^2 + \mathcal{O}(\sigma^4)
\end{aligned}
$$

Proof. There exists σ and $\boldsymbol{\xi}$ small enough that first-order Taylor series expansion holds true

$$
\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x} + \boldsymbol{\xi}) \Big)^2
$$

$$
\begin{aligned}
&= \mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}) - \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \mathcal{S}^i(\mathbf{x}) \Big)^2 && \text{(Taylor series)} \\
&+ \mathcal{O}(\sigma^4) \\
&= \sum_{i=1}^{k} \Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}) \Big)^2 && \text{(expand the square)} \\
&+ \mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \Big[\boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \mathcal{S}^i(\mathbf{x}) \Big]^2 \\
&- \mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \mathcal{S}^i(\mathbf{x}) [y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})] && \text{(=0, as } \boldsymbol{\xi} \text{ is zero mean)} \\
&+ \mathcal{O}(\sigma^4) \\
&= \sum_{i=1}^{k} \Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}) \Big)^2 \\
&+ \sum_{i=1}^{k} \nabla_{\mathbf{x}} \mathcal{S}^i(\mathbf{x})^\top \mathbb{E}_{\boldsymbol{\xi}} \boldsymbol{\xi} \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \mathcal{S}^i(\mathbf{x}) && \text{(linearity of expectation)} \\
&+ \mathcal{O}(\sigma^4) \\
&= \sum_{i=1}^{k} \Big(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}) \Big)^2 \\
&+ \sum_{i=1}^{k} \sigma^2 \|\nabla_{\mathbf{x}} \mathcal{S}^i(\mathbf{x})\|^2 && (\mathbb{E}_{\boldsymbol{\xi}} \boldsymbol{\xi} \boldsymbol{\xi}^\top = \sigma I) \\
&+ \mathcal{O}(\sigma^4)
\end{aligned}
$$

□

Proposition. *Consider the **cross-entropy** cost function for training a neural network with k targets, given by* $\ell(y(\mathbf{x}), \mathcal{S}(\mathbf{x})) = \sum_{i=1}^{k} -y^i(\mathbf{x}) \log \mathcal{S}^i(\mathbf{x})$*, where* $\mathbf{x} \in \mathbb{R}^D$ *is an input data point, and* $y^i(\mathbf{x})$ *is the* i^{th} *target output. Let* $\boldsymbol{\xi}$ $(\in \mathbb{R}^D) = \sigma$ $\mathbf{z}$ *be a scaled version of a unit normal random variable* $\mathbf{z}$ $\in \mathbb{R}^D$ *with scaling factor* $\sigma \in \mathbb{R}$*. Then the following is true.*

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}} \left[-\sum_{i=1}^{k} y^i(\mathbf{x}) \log(\mathcal{S}_s^i(\mathbf{x} + \boldsymbol{\xi})) \right] \\
&\approx -\sum_{i=1}^{k} y^i(\mathbf{x}) \log(\mathcal{S}_s^i(\mathbf{x})) \;+\; \sigma^2 \sum_{i=1}^{k} y^i(\mathbf{x}) \frac{\|\nabla_x \mathcal{S}_s^i(\mathbf{x})\|_2^2}{\mathcal{S}_s^i(\mathbf{x})^2}
\end{aligned}
\tag{2.10}
$$

Proof. There exists σ and $\boldsymbol{\xi}$ small enough that first-order Taylor series expansion holds true

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}} - \sum_{i=1}^{k} y^i(\mathbf{x}) \log \mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi}) \\
=&\mathbb{E}_{\boldsymbol{\xi}} - \sum_{i=1}^{k} y^i(\mathbf{x}) \left(\log \mathcal{S}^i(\mathbf{x}) + \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) + \frac{1}{2} \boldsymbol{\xi}^\top \nabla^2_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) \boldsymbol{\xi} \right) && \text{(Taylor series)} \\
&-\mathcal{O}(\sigma^3) \\
=&-\sum_{i=1}^{k} y^i(\mathbf{x}) \log \mathcal{S}^i(\mathbf{x}) \\
&-\frac{1}{2} \mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} y^i(\mathbf{x}) \boldsymbol{\xi}^\top \nabla^2_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) \boldsymbol{\xi} \\
&-\mathbb{E}_{\boldsymbol{\xi}} \sum_{i=1}^{k} y^i(\mathbf{x}) \boldsymbol{\xi}^\top \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) && \text{(=0, as } \boldsymbol{\xi} \text{ is zero mean)} \\
&-\mathcal{O}(\sigma^4) \\
=&-\sum_{i=1}^{k} \mathcal{T}^i(\mathbf{x}) \log \mathcal{S}^i(\mathbf{x}) \\
&-\frac{1}{2} \sum_{i=1}^{k} y^i(\mathbf{x}) \mathbb{E}_{\boldsymbol{\xi}} \boldsymbol{\xi}^\top \nabla^2_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) \boldsymbol{\xi} && \text{(linearity of expectation)} \\
&-\mathcal{O}(\sigma^4)
\end{aligned}
$$

Now, we approximate the Hessian term with the Fisher Information matrix, which is a common trick used to simplify Hessian computations. Specifically we have, $\nabla^2_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) = \nabla^2_{\mathbf{x}} \mathcal{S}^i(\mathbf{x}) / \mathcal{S}^i(\mathbf{x}) - \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x})^\top \approx -\nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x}) \nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x})^\top$. This assumes that $\nabla^2_{\mathbf{x}} \mathcal{S}^i(\mathbf{x}) \approx 0$ which is true for pre-softmax logits of ReLU nets but not necessarily for post-softmax probabilities. Using this approximation, we have

$$
\begin{aligned}
&\approx -\sum_{i=1}^{k} \mathcal{T}^i(\mathbf{x}) \log \mathcal{S}^i(\mathbf{x}) \\
&+\frac{1}{2} \sum_{i=1}^{k} y^i(\mathbf{x}) \mathbb{E}_{\boldsymbol{\xi}} (\nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x})^\top \boldsymbol{\xi})^2 && \text{(Fisher approximation)} \\
&\approx -\sum_{i=1}^{k} \mathcal{T}^i(\mathbf{x}) \log \mathcal{S}^i(\mathbf{x}) \\
&+\frac{\sigma^2}{2} \sum_{i=1}^{k} y^i(\mathbf{x}) \|\nabla_{\mathbf{x}} \log \mathcal{S}^i(\mathbf{x})\|^2
\end{aligned}
$$

□

2.10 Proof of Proposition 3

Proposition. *From the notations in the main text, we have*

$$\frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}\sim\mathcal{D}_s}\ell(f(\mathbf{x}),g(\mathbf{x}))\leq\max_{\mathbf{x}\sim\mathcal{D}_t}\ell(f(\mathbf{x}),g(\mathbf{x}))+\mathrm{K}\mathcal{H}_a(\mathcal{D}_s,\mathcal{D}_t)$$

Proof. Let us denote $\rho(\mathbf{x})=\ell(f(\mathbf{x}),g(\mathbf{x}))$ for convenience. Assume Lipschitz continuity for $\rho(\mathbf{x})$ with Lipschitz constant K, and distance metric $\psi_{\mathbf{x}}(\cdot,\cdot)$ in the input space -

$$\begin{aligned}\|\rho(\mathbf{x}_1)-\rho(\mathbf{x}_2)\|_1&\leq\mathrm{K}\psi_{\mathbf{x}}(\mathbf{x}_1,\mathbf{x}_2)\\ \implies\rho(\mathbf{x}_1)&\leq\rho(\mathbf{x}_2)+\mathrm{K}\psi_{\mathbf{x}}(\mathbf{x}_1,\mathbf{x}_2)\end{aligned}$$

Assuming that $\rho(\mathbf{x}_1)\geq\rho(\mathbf{x}_2)$. Note that it holds even otherwise, but is trivial.

Now, for every datapoint $\mathbf{x}_s\in\mathcal{D}_s$, there exists a point $\mathbf{x}_t\in\mathcal{D}_t$ such that $\psi_{\mathbf{x}}(\mathbf{x}_t,\mathbf{x}_s)$ is the smallest among all points in $\mathcal{D}_t$. In other words, we look at the point in $\mathcal{D}_t$ closest to each point $\mathbf{x}_s$. Note that in this process only a subset of points $\mathrm{d}_t\subseteq\mathcal{D}_t$ are chosen, and individual points can be chosen multiple times. For these points, we can write

$$\begin{aligned}&\rho(\mathbf{x}_s)\leq\rho(\mathbf{x}_t)+\mathrm{K}\psi_{\mathbf{x}}(\mathbf{x}_t,\mathbf{x}_s)\\ \implies\frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}_s\sim\mathcal{D}_s}\rho(\mathbf{x}_s)&\leq\frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}_t\ \text{closest to}\ \mathbf{x}_s}\rho(\mathbf{x}_t)\\ &\quad+\frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}_t\ \text{closest to}\ \mathbf{x}_s}\mathrm{K}\psi_{\mathbf{x}}(\mathbf{x}_t,\mathbf{x}_s)\end{aligned}$$

We see that $\frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}_t}\rho(\mathbf{x}_t)\leq\max_{\mathbf{x}\sim\mathrm{d}_t}\rho(\mathbf{x})\leq\max_{\mathbf{x}\sim\mathcal{D}_t}\rho(\mathbf{x})$, which is a consequence of the fact that the max is greater than any convex combination of elements.

Also, we have $\psi_{\mathbf{x}}(\mathbf{x}_s,\mathbf{x}_t)\leq\mathcal{H}_a(\mathcal{D}_s,\mathcal{D}_t)$, which is the maximum distance between any two 'closest' points from $\mathcal{D}_s$ to $\mathcal{D}_t$ (by definition).

Applying these bounds, we have the final result.

□

2.10.1 Proof for Corollary

Corollary. *For any superset* $\mathrm{D'}_t \supseteq \mathcal{D}_t$ *of the target dataset,* $\mathcal{H}_a(\mathcal{D}_s, \mathcal{D'}_t) \leq \mathcal{H}_a(\mathcal{D}_s, \mathcal{D}_t)$

Proof. From the previous proof, we have $\rho(\mathbf{x}_s) \leq \rho(\mathbf{x}_t) + \mathrm{K}\psi_{\mathbf{x}}(\mathbf{x}_t, \mathbf{x}_s)$ for an individual point $\mathbf{x}_s$. Now if we have $\mathrm{D'}_t \supseteq \mathcal{D}_t$, then we have $\rho(\mathbf{x}_s) \leq \rho(\mathbf{x}'_t) + \mathrm{K}\psi_{\mathbf{x}}(\mathbf{x}'_t, \mathbf{x}_s)$, where $\mathbf{x}'_t$ is the new point closest to $\mathbf{x}_s$. It is clear that $\psi_{\mathbf{x}}(\mathbf{x}'_t, \mathbf{x}_s) \leq \psi_{\mathbf{x}}(\mathbf{x}_t, \mathbf{x}_s)$ for all $\mathbf{x}_s$. Hence it follows that $\mathcal{H}_a(\mathcal{D}_s, \mathcal{D'}_t) \leq \mathcal{H}_a(\mathcal{D}_s, \mathcal{D}_t)$.

□

2.11 Justification for gradient loss

We use the following loss term for gradient matching for transfer learning.

$$\text{Match gradients} = \left|\left| \frac{\nabla_x f(\mathbf{x})}{\|\nabla_x f(\mathbf{x})\|_2} - \frac{\nabla_x g(\mathbf{x})}{\|\nabla_x g(\mathbf{x})\|_2} \right|\right|_2^2 \tag{2.11}$$

We can show that the above loss term corresponds to adding a noise term $\boldsymbol{\xi}_f \propto \|\nabla_x f(\mathbf{x})\|_2^{-1}$ for $f(\mathbf{x})$ and $\boldsymbol{\xi}_g \propto \|\nabla_x g(\mathbf{x})\|_2^{-1}$ for $g(\mathbf{x})$ for the distillation loss. From the first order Taylor series expansion, we see that $g(x + \boldsymbol{\xi}) = g(x) + \boldsymbol{\xi}_g \nabla_x g(\mathbf{x})$. Thus for networks $f(\cdot)$ and $g(\cdot)$ with different gradient magnitudes, we expect different responses for the same noisy inputs. Specifically, we see that $\mathbb{E}_{\boldsymbol{\xi}_g}\|g(x + \boldsymbol{\xi}_g) - g(x)\|_2^2 = \sigma_g^2 \|\nabla_x g(\mathbf{x})\|_2^2 = \sigma^2 \frac{\|\nabla_x g(\mathbf{x})\|_2^2}{\|\nabla_x g(\mathbf{x})\|_2^2} = \sigma^2$ for a gaussian model with covariance matrix being σ times the identity.

2.12 Experimental details

2.12.1 VGG Network Architectures

The architecture for our networks follow the VGG design philosophy. Specifically, we have blocks with the following elements:

- 3×3 conv kernels with c channels of stride 1
- Batch Normalization
- ReLU

Whenever we use Max-pooling (M), we use stride 2 and window size 2.

The architecture for VGG-9 is - $[64 - M - 128 - M - 256 - 256 - M - 512 - 512 - M - 512 - 512 - M]$. Here, the number stands for the number of convolution channels, and M represents max-pooling. At the end of all the convolutional and max-pooling layers, we have a Global Average Pooling (GAP) layer, after which we have a fully connected layer leading up to the final classes. Similar architecture is used for the case of both CIFAR and MIT Scene experiments.

The architecture for VGG-4 is - $[64 - M - 128 - M - 512 - M]$.

2.12.2 Loss function

The loss function for distillation experiments use the following form.

$$\ell(\mathcal{S}, \mathcal{T}) = \alpha \times (\text{CE}) + \beta \times (\text{Match Activations}) + \gamma \times (\text{Match gradients})$$

In our experiments, α, β, γ are either set to 1 or 0. In other words, all regularization constants are 1.

Here, 'CE' refers to cross-entropy with ground truth labels. 'Match Activations' refers to squared error term over pre-softmax activations of the form $(y_s - y_t)^2$. 'Match gradients' refers to the same squared error term, but for gradients.

For the MIT Scene experiments, α, β, γ are either set to 10 or 0, depending on the specific method. To compute the gradient, we use average pooling over a $feature\ size/5$ window with a stride of 1. We match the gradient after the first residual block for resnet, and after the second max-pool for VGG. This corresponds to feature level "1" in the ablation experiments.

2.12.3 Optimization

For CIFAR100 experiments, we run optimization for 500 epochs. We use the Adam optimizer, with an initial learning rate of $1e-3$, and a single learning rate annealing (to $1e-4$) at 400 epochs. We used a batch size of 128.

For MIT Scenes, we used SGD with momentum of 0.9, for 75 epochs. The initial learning rate is $1e-3$, and it is reduced 10 times after 40 and 60 epochs. We used batch size 8. This is because the gradient computation is very memory intensive.

3 Full-Gradient Representation for Neural Network Visualization

We introduce a new tool for interpreting neural net responses, namely full-gradients, which decomposes the neural net response into input sensitivity and per-neuron sensitivity components. This is the first proposed representation which satisfies two key properties: completeness and weak dependence, which provably cannot be satisfied by any saliency map-based interpretability method. For convolutional nets, we also propose an approximate saliency map representation, called FullGrad, obtained by aggregating the full-gradient components. We experimentally evaluate the usefulness of FullGrad in explaining model behaviour with two quantitative tests: pixel perturbation and remove-and-retrain. Our experiments reveal that our method explains model behaviour correctly, and more comprehensively, than other methods in the literature. Visual inspection also reveals that our saliency maps are sharper and more tightly confined to object regions than other methods.

3.1 Introduction

This work studies saliency map representations for the interpretation of neural network functions. Saliency maps assign to each input feature an importance score, which is a measure of the usefulness of that feature for the task performed by the neural network. However, the presence of internal structure among features sometimes makes it difficult to assign a single importance score per feature. For example, input spaces such as that of natural images are compositional in nature. This means that while any single individual pixel in an image may be unimportant on its own, a collection of pixels may be critical if they form an important image region such as an object part.

For example, a bicycle in an image can still be identified if any single pixel is missing, but if the entire collection of pixels corresponding to a key element, such as a wheel or the drive chain, are missing, then it becomes much more difficult. Here the importance of a part cannot be deduced from the individual importance of its constituent pixels, as

each such individual pixel is unimportant on its own. An ideal interpretability method would not just provide importance for each pixel, but also capture that of groups of pixels which have an underlying structure.

This tension also reveals itself in the formal study of saliency maps. While there is no single formal definition of saliency, there are several intuitive characteristics that the community has deemed important (Sundararajan et al., 2017; Montavon et al., 2017; Shrikumar et al., 2017; Lundberg and Lee, 2017; Kindermans et al., 2017; Adebayo et al., 2018). One such characteristic is that an input feature must be considered important if changes to that feature greatly affect the neural network output (Kindermans et al., 2017; Simonyan et al., 2013). Another desirable characteristic is that the saliency map must completely explain the neural network output, i.e., the individual feature importance scores must add up to the neural network output (Sundararajan et al., 2017; Montavon et al., 2017; Shrikumar et al., 2017). This is done by a redistribution of the numerical output score to individual input features. In this view, a feature is important if it makes a large numerical contribution to the output. Thus we have two distinct notions of feature importance, both of which are intuitive. The first notion of importance assignment is called *local* attribution and second, *global* attribution. It is almost always the case for practical neural networks that these two notions yield methods that consider entirely different sets of features to be important, which is counter-intuitive.

In this Chapter, we propose full-gradients, a representation which assigns importance scores to both the input features and individual feature detectors (or neurons) in a neural network. Input attribution helps capture importance of individual input pixels, while neuron importances capture importance of groups of pixels, accounting for their structure. In addition, full-gradients achieve this by simultaneously satisfying both notions of *local* and *global* importance. To the best of our knowledge, no previous method in literature has this property.

Our contributions here are:

1. We show in § 3.3 that *weak dependence* (see Definition 1), a notion of local importance, and *completeness* (see Definition 2), a notion of global importance, cannot be satisfied simultaneously by any saliency method. This suggests that the counter-intuitive behavior of saliency methods reported in literature (Shrikumar et al., 2017; Kindermans et al., 2017) is unavoidable.

2. We introduce in § 3.4 the full-gradients which are more expressive than saliency maps, and satisfy both importance notions simultaneously. We also use this to define approximate saliency maps for convolutional nets, dubbed FullGrad, by leveraging strong geometric priors induced by convolutions.

3. We perform in § 5.5 quantitative tests on full-gradient saliency maps including pixel perturbation and remove-and-retrain (Hooker et al., 2018), which show that

FullGrad outperforms existing competitive methods.

3.2 Related Work

Within the vast literature on interpretability of neural networks, we shall restrict discussion solely to saliency maps or input attribution methods. First attempts at obtaining saliency maps for modern deep networks involved using input-gradients (Simonyan et al., 2013) and deconvolution (Zeiler and Fergus, 2014). Guided backprop (Springenberg et al., 2014) is another variant obtained by changing the backprop rule for input-gradients to produce cleaner saliency maps. Recent works have also adopted axiomatic approaches to attribution by proposing methods that explicitly satisfy certain intuitive properties. Deep Taylor decomposition (Montavon et al., 2017), DeepLIFT (Shrikumar et al., 2017), Integrated gradients (Sundararajan et al., 2017) and DeepSHAP (Lundberg and Lee, 2017) adopt this broad approach. Central to all these approaches is the requirement of *completeness* which requires that the saliency map account for the function output in an exact numerical sense. In particular, Lundberg and Lee (2017) and Ancona et al. (2018) propose unifying frameworks for several of these saliency methods.

However, some recent work also shows the fragility of some of these methods. These include unintuitive properties such as being insensitive to model randomization (Adebayo et al., 2018), partly recovering the input (Nie et al., 2018) or being insensitive to the model's invariances (Kindermans et al., 2017). One possible reason attributed for the presence of such fragilities is evaluation of attribution methods, which are often solely based on visual inspection. As a result, need for quantitative evaluation methods is urgent. Popular quantitative evaluation methods in literature are based on image perturbation (Samek et al., 2016; Ancona et al., 2018; Montavon et al., 2017). These tests broadly involve removing the most salient pixels in an image, and checking whether they affect the neural network output. However, removing pixels can cause artifacts to appear in images. To compensate for this, RemOve And Retrain (ROAR) (Hooker et al., 2018) propose a retraining-based procedure. However, this method too has drawbacks as retraining can cause the model to focus on parts of the input it had previously ignored, thus not explaining the original model. Hence we do not yet have completely rigorous methods for saliency map evaluation.

Similar to our work, some works (Shrikumar et al., 2017; Kindermans et al., 2016) also make the observation that including biases within attributions can enable gradient-based attributions to satisfy the completeness property. However, they do not propose attribution methods based on this observation like we do here.

3.3 Local *vs.* Global Attribution

In this section, we show that there cannot exist saliency maps that satisfy both notions of *local* and *global* attribution. We do this by drawing attention to a simple fact that $D-$dimensional saliency map cannot summarize even linear models in $\mathbb{R}^D$, as such linear models have $D+1$ parameters. We prove our results by defining a weak notion of local attribution which we call *weak dependence*, and a weak notion of global attribution, called *completeness*.

Let us consider a neural network function $f : \mathbb{R}^D \to \mathbb{R}$ with inputs $\mathbf{x} \in \mathbb{R}^D$. A saliency map $S(\mathbf{x}) = \sigma(f, \mathbf{x}) \in \mathbb{R}^D$ is a function of the neural network f and an input $\mathbf{x}$. For linear models of the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, it is common to visualize the weights $\mathbf{w}$. For this case, we observe that the saliency map $S(\mathbf{x}) = \mathbf{w}$ is independent of $\mathbf{x}$. Similarly, piecewise-linear models can be thought of as collections of linear models, with each linear model being defined on a different local neighborhood. For such cases, we can define weak dependence as follows.

Definition 1. *(Weak dependence on inputs) Consider a piecewise-linear model*

$$f(\mathbf{x}) = \begin{cases} \mathbf{w}_0^T\mathbf{x} + b_0 & \mathbf{x} \in \mathcal{U}_0 \\ ... & \\ \mathbf{w}_n^T\mathbf{x} + b_n & \mathbf{x} \in \mathcal{U}_n \end{cases}$$

where all $\mathcal{U}_i$ are open connected sets. For this function, the saliency map $S(\mathbf{x}) = \sigma(f, \mathbf{x})$ restricted to a set $\mathcal{U}_i$ is independent of $\mathbf{x}$, and depends only on the parameters $\mathbf{w}_i, b_i$.

Hence in this case $S(\mathbf{x})$ depends weakly on $\mathbf{x}$ by being dependent only on the neighborhood $\mathcal{U}_i$ in which $\mathbf{x}$ resides. This generalizes the notion of *local* importance to piecewise-linear functions. A stronger form of this property, called *input invariance*, was deemed desirable in previous work (Kindermans et al., 2017), which required saliency methods to mirror model sensitivity. Methods which satisfy our weak dependence include input-gradients (Simonyan et al., 2013), guided-backprop (Springenberg et al., 2014) and deconv (Zeiler and Fergus, 2014). Note that our definition of weak dependence also allows for two disconnected sets having the same linear parameters $(\mathbf{w}_i, b_i)$ to have different saliency maps, and hence in that sense is more general than input invariance (Kindermans et al., 2017), which does not allow for this. We now define completeness for a saliency map by generalizing equivalent notions presented in prior work (Sundararajan et al., 2017; Montavon et al., 2017; Shrikumar et al., 2017).

Definition 2. *(Completeness) A saliency map $S(\mathbf{x})$ is*

- *complete if there exists a function ϕ such that $\phi(S(\mathbf{x}), \mathbf{x}) = f(\mathbf{x})$ for all $f, \mathbf{x}$.*

- *complete with a baseline* $\mathbf{x}_0$ *if there exists a function* ϕ_c *such that* $\phi_c(S(\mathbf{x}), S_0(\mathbf{x}_0), \mathbf{x}, \mathbf{x}_0) = f(\mathbf{x}) - f(\mathbf{x}_0)$ *for all* $f, \mathbf{x}, \mathbf{x}_0$*, where* $S_0(\mathbf{x}_0)$ *is the saliency map of* $\mathbf{x}_0$*.*

The intuition here is that if we expect $S(\mathbf{x})$ to completely encode the computation performed by f, then it must be possible to recover $f(\mathbf{x})$ by using the saliency map $S(\mathbf{x})$ and input $\mathbf{x}$. Note that the second definition is more general, and in principle subsumes the first. We are now ready to state our impossibility result.

Proposition 4. *For any piecewise-linear function* f*, it is impossible to obtain a saliency map* S *that satisfies both completeness and weak dependence on inputs, in general.*

The proof is provided in the supplementary material. A natural consequence of this is that methods such as integrated gradients (Sundararajan et al., 2017), deep Taylor decomposition (Montavon et al., 2017) and DeepLIFT (Shrikumar et al., 2017) which satisfy completeness do not satisfy weak dependence. For the case of integrated gradients, we provide a simple illustration showing how this can lead to unintuitive attributions. Given a baseline $\mathbf{x}'$, integrated gradients (IG) is given by $\text{IG}_i(\mathbf{x}) = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial f(\mathbf{x}' + \alpha(\mathbf{x} - \mathbf{x}'))}{\partial x_i} \text{d}\alpha$, where x_i is the i^{th} input co-ordinate.

Example 1. *(Integrated gradients (Sundararajan et al., 2017) can be counter-intuitive)*

Consider the piecewise-linear function for inputs $(x_1, x_2) \in \mathbb{R}^2$*.*

$$f(x_1, x_2) = \begin{cases} x_1 + 3x_2 & x_1, x_2 \leq 1 \\ 3x_1 + x_2 & x_1, x_2 > 1 \\ 0 & otherwise \end{cases}$$

Assume baseline $\mathbf{x}' = (0, 0)$*. Consider three points* $(2, 2), (4, 4), (1.5, 1.5)$ *, all of which satisfy* $x_1, x_2 > 1$ *and thus are subject to the same linear function of* $f(x_1, x_2) = 3x_1 + x_2$*. However, depending on which point we consider, IG yields different relative importances among the input features. E.g:* $IG(x_1 = 4, x_2 = 4) = (10, 6)$ *where it seems that* x_1 *is more important (as* $10 > 6$*), while for* $IG(1.5, 1.5) = (2.5, 3.5)$*, it seems that* x_2 *is more important. Further, at* $IG(2, 2) = (4, 4)$ *both co-ordinates are assigned equal importance. However in all three cases, the output is clearly more sensitive to changes to* x_1 *than it is to* x_2 *as they lie on* $f(x_1, x_2) = 3x_1 + x_2$*, and thus attributions to* $(2, 2)$ *and* $(1.5, 1.5)$ *are counter-intuitive.*

Thus it is clear that two intuitive properties of weak dependence and completeness cannot be satisfied simultaneously. Both are intuitive notions for saliency maps and thus satisfying just one makes the saliency map counter-intuitive by not satisfying the other. Similar counter-intuitive phenomena observed in literature may be unavoidable. For example, Shrikumar et al. (2017) show counter-intuitive behavior of local attribution

methods by invoking a property similar global attribution, called *saturation sensitivity*. On the other hand, Kindermans et al. (2017) show fragility for global attribution methods by appealing to a property similar to local attribution, called *input insensitivity*.

This paradox occurs primarily because saliency maps are too restrictive, as both weights and biases of a linear model cannot be summarized by a saliency map. While exclusion of the bias term in linear models to visualize only the weights seems harmless, the effect of such exclusion compounds rapidly for neural networks which have bias terms for each neuron. Neural network biases cannot be collapsed to a constant scalar term like in linear models, and hence cannot be excluded. In the next section we shall look at full-gradients, which is a more expressive tool than saliency maps, accounts for bias terms and satisfies both weak dependence and completeness.

3.4 Full-Gradient Representation

In this section, we introduce the full-gradient representation, which provides attribution to both inputs and neurons. We proceed by observing the following result for ReLU networks.

Proposition 5. *Let f be a ReLU neural network without bias parameters, then $f(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x})^T \mathbf{x}$.*

The proof uses the fact that for such nets, $f(k\mathbf{x}) = kf(\mathbf{x})$ for any $k > 0$. This can be extended to ReLU neural networks with bias parameters by incorporating additional inputs for biases, which is a standard trick used for the analysis of linear models. For a ReLU network $f(\cdot; \mathbf{b})$ with bias, let the number of such biases in f be F.

Proposition 6. *Let f be a ReLU neural network with biases $\mathbf{b} \in \mathbb{R}^F$, then*

$$f(\mathbf{x}; \mathbf{b}) = \nabla_{\mathbf{x}} f(\mathbf{x}; \mathbf{b})^T \mathbf{x} + \nabla_b f(\mathbf{x}; \mathbf{b})^T \mathbf{b} \tag{3.1}$$

The proof for these statements is provided in the supplementary material. Here biases include both explicit bias parameters and well as implicit biases, such as running averages of batch norm layers. For practical networks, we have observed that these implicit biases are often much larger in magnitude than explicit bias parameters, and hence might be more important.

We can extend this decomposition to non-ReLU networks by considering implicit biases arising due to usage of generic non-linearities. For this, we linearize a non-linearity $y = \sigma(x)$ at a neighborhood around x to obtain $y = \frac{\mathrm{d}\sigma(x)}{\mathrm{d}x} x + b_\sigma$. Here b_σ is the implicit bias that is unaccounted for by the derivative. Note that for ReLU-like non-linearities, $b_\sigma = 0$. As a result, we can trivially extend the representation to arbitrary non-linearities by appending b_σ to the vector $\mathbf{b}$ of biases. In general, *any* quantity that is unaccounted

for by the input-gradient is an implicit bias, and thus by definition, together they must add up to the function output, like in equation 3.3.

Equation 3.3 is an alternate representation of the neural network output in terms of various gradient terms. We shall call $\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{b})$ as input-gradients, and $\nabla_b f(\mathbf{x}, \mathbf{b}) \odot \mathbf{b}$ as the bias-gradients. Together, they constitute full-gradients. To the best our knowledge, this is the only other exact representation of neural network outputs, other than the usual feed-forward neural net representation in terms of weights and biases.

For the rest of the Chapter, we shall henceforth use the shorthand notation $f^b(\mathbf{x})$ for $\nabla_b f(\mathbf{x}, \mathbf{b}) \odot \mathbf{b}$, the bias-gradient, and drop the explicit dependence on $\mathbf{b}$ in $f(\mathbf{x}, \mathbf{b})$.

3.4.1 Properties of Full-Gradients

Here discuss some intuitive properties of full-gradients. We shall assume that full-gradients comprise of the pair $G = (\nabla_x f(\mathbf{x}), f^b(\mathbf{x})) \in \mathbb{R}^{D+F}$. We shall also assume with no loss of generality that the network contains ReLU non-linearity without batch-norm, and that all biases are due to bias parameters.

Weak dependence on inputs: For a piecewise linear function f, it is clear that the input-gradient is locally constant in a linear region. It turns out that a similar property holds for $f^b(\mathbf{x})$ as well, and a short proof of this can be found in the supplementary material.

Completeness: From equation 3.3, we see that the full-gradients exactly recover the function output $f(\mathbf{x})$, satisfying completeness.

Saturation sensitivity: Broadly, saturation refers to the phenomenon of zero input attribution to regions of zero function gradient. This notion is closely related to global attribution, as it requires saliency methods to look beyond input sensitivity. As an example used in prior work (Sundararajan et al., 2017), consider $f(x) = a - \text{ReLU}(b - x)$, with $a = b = 1$. At $x = 2$, even though $f(x) = 1$, the attribution to the only input is zero, which is deemed counter-intuitive. Integrated gradients (Sundararajan et al., 2017) and DeepLIFT (Shrikumar et al., 2017) consider handling such saturation for saliency maps to be a central issue and introduce the concept of baseline inputs to tackle this. However, one potential issue with this is that the attribution to the input now depends on the choice of baseline for a given function. To avoid this, we here argue that is better to also provide attributions to some function parameters. In the example shown, the function $f(x)$ has two biases (a, b) and the full-gradient method attributes $(1, 0)$ to these biases for input $x = 2$.

Full Sensitivity to Function Mapping: Adebayo et al. (2018) recently proposed sanity check criteria that every saliency map must satisfy. The first of these criteria

is that a saliency map must be sensitive to randomization of the model parameters. Random parameters produce incorrect input-output mappings, which must be reflected in the saliency map. The second sanity test is that saliency maps must change if the data used to train the model have their labels randomized. A stronger criterion which generalizes both these criteria is that saliency maps must be sensitive to any change in the function mapping, induced by changing the parameters. This change of parameters can occur by either explicit randomization of parameters or training with different data. It turns out that input-gradient based methods are insensitive to some bias parameters as shown below.

Example 2. *(Bias insensitivity of input-gradient methods)*

Consider a one-hidden layer net of the form $f(x) = w_1 * \text{relu}(w_0 * x + b_0) + b_1$. *For this, it is easy to see that input-gradients (Simonyan et al., 2013) are insensitive to small changes in* b_0 *and arbitrarily large changes in* b_1. *This applies to all input-gradient methods such as guided backprop (Springenberg et al., 2014) and deconv (Zeiler and Fergus, 2014). Thus none of these methods satisfy the model randomization test on* $f(x)$ *upon randomizing* b_1.

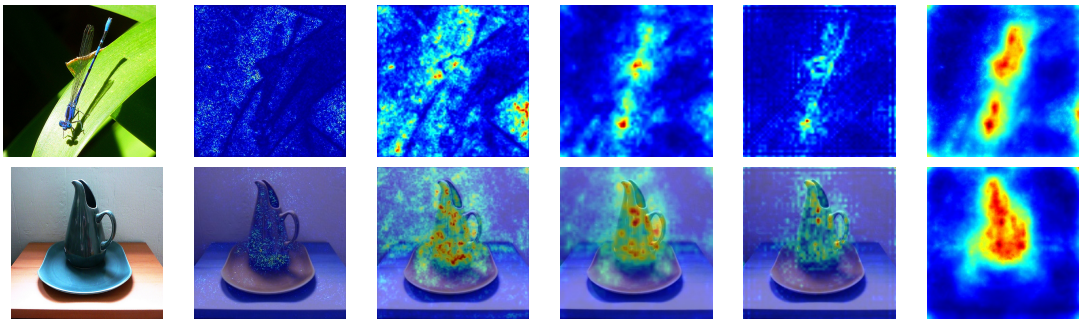
Note that in this example, only the sensitivity to the model biases in the first layer are considered, and not model weights, which was the subject of study in Adebayo et al. (2018).

On the other hand, full-gradients are sensitive to every parameter that affects the function mapping. In particular, by equation 3.3 we observe that given full-gradients G, we have $\frac{\partial G}{\partial \theta_i} = 0$ for a parameter θ_i, if and only if $\frac{\partial f}{\partial \theta_i} = 0$.

3.4.2 FullGrad: Full-Gradient Saliency Maps for Convolutional Nets

For convolutional networks, bias-gradients have a spatial structure which is convenient to visualize. Consider a single convolutional filter $\mathbf{z} = \mathbf{w} * \mathbf{x} + \mathbf{b}$ where $\mathbf{w} \in \mathbb{R}^{2k+1}$, $\mathbf{b} = [b, b....b] \in R^D$ and $(*)$ for simplicity refers to a convolution with appropriate padding applied so that $\mathbf{w}*\mathbf{x} \in \mathbb{R}^D$, which is often the case with practical convolutional nets. Here the bias parameter is a single scalar b repeated D times due to the weight sharing nature of convolutions. For this particular filter, the bias-gradient $f^b(\mathbf{x}) = \nabla_{\mathbf{z}} f(\mathbf{x}) \odot \mathbf{b} \in \mathbb{R}^D$ is shaped like the input $\mathbf{x} \in \mathbb{R}^D$, and hence can be visualized like the input. Further, the locally connected nature of convolutions imply that each co-ordinate $f^b(\mathbf{x})_i$ is a function of only $\mathbf{x}[i-k, i+k]$, thus capturing the importance of a group of input co-ordinates centered at i. This is easily ensured for practical convolutional networks (e.g.: VGG, ResNet, DenseNet, etc) which are often designed such that feature sizes of immediate layers match and are aligned by appropriate padding.

For such nets we can now visualize per-neuron and per-layer maps using bias-gradients.



Input | Input-grad × input | Layer 3 bias-gradient | Layer 5 bias-gradient | Layer 7 bias-gradient | **FullGrad aggregate**

Figure 3.1 – Visualization of bias-gradients at different layers of a VGG-16 pre-trained neural network. While none of the intermediate layer bias-gradients themselves demarcate the object satisfactorily, the full-gradient map achieves this by aggregating information from the input-gradient and all intermediate bias-gradients. (see Equation 3.2).

Per-neuron maps are obtained by visualizing a spatial map $\in \mathbb{R}^D$ for every convolutional filter. Per-layer maps are obtained by aggregating such neuron-wise maps. An example is shown in Figure 3.1. For images, we visualize these maps after performing standard post-processing steps that ensure good viewing contrast. These post-processing steps are simple re-scaling operations, often supplemented with an absolute value operation to visualize only the magnitude of importance while ignoring the sign. One can also visualize separately the positive and negative parts of the map to avoid ignoring signs. Let such post-processing operations be represented by $\psi(\cdot)$. For maps that are downscaled versions of inputs, such post-processing also includes a resizing operation, often done by standard algorithms such as cubic interpolation.

We can also similarly visualize approximate network-wide saliency maps by aggregating such layer-wise maps. Let c run across channels c_l of a layer l in a neural network, then the FullGrad saliency map $S_f(\mathbf{x})$ is given by

$$S_f(\mathbf{x}) = \psi(\nabla_{\mathbf{x}} f(\mathbf{x}) \odot \mathbf{x}) + \sum_{l \in L} \sum_{c \in c_l} \psi\left(f^b(\mathbf{x})_c\right) \tag{3.2}$$

Here, $\psi(\cdot)$ is the post-processing operator discussed above. For this work, we choose $\psi(\cdot) =$ `bilinearUpsample(rescale(abs(·)))`, where `rescale(·)` linearly rescales values to lie between 0 and 1, and `bilinearUpsample(·)` upsamples the gradient maps using bilinear interpolation to have the same spatial size as the image. For a network with both convolutional and fully-connected layers, we can obtain spatial maps for only the convolutional layers and hence the effect of fully-connected layers' bias parameters are not completely accounted for. Note that omitting $\psi(\cdot)$ and performing an additional

spatial aggregation in the equation above results in the exact neural net output value according to the full-gradient decomposition. Further discussion on post-processing is presented in Section 3.6.

We stress here that the FullGrad saliency map described here is approximate, in the sense that the full representation is in fact $G = (\nabla_x f(\mathbf{x}), f^b(\mathbf{x})) \in \mathbb{R}^{D+F}$, and our network-wide saliency map merely attempts to capture information from multiple maps into a single visually coherent one. This saliency map has the disadvantage that all saliency maps have, i.e. they cannot satisfy both completeness and weak dependence at the same time, and changing the aggregation method (such as removing $\odot\mathbf{x}$ in equation 3.2, or changing $\psi(\cdot)$) can help us satisfy one property or the other. Experimentally we find that aggregating maps as per equation 3.2 produces the sharpest maps, as it enables neuron-wise maps to vote independently on the importance of each spatial location.

3.5 Experiments

To show the effectiveness of FullGrad, we perform two quantitative experiments. First, we use a pixel perturbation procedure to evaluate saliency maps on the Imagenet 2012 dataset. Second, we use the remove and retrain procedure (Hooker et al., 2018) to evaluate saliency maps on the CIFAR100 dataset.

3.5.1 Pixel perturbation

Popular methods to benchmark saliency algorithms are variations of the following procedure: remove k most salient pixels and check variation in function value. The intuition is that good saliency algorithms identify pixels that are important to classification and hence cause higher function output variation. Benchmarks with this broad strategy are employed in Samek et al. (2016); Ancona et al. (2018). However, this is not a perfect benchmark because replacing image pixels with black pixels can cause high-frequency edge artifacts to appear which may cause output variation. When we employed this strategy for a VGG-16 network trained on Imagenet, we find that several saliency methods have similar output variation to random pixel removal. This effect is also present in large scale experiments (Samek et al., 2016; Ancona et al., 2018). This occurs because random pixel removal creates a large number of disparate artifacts that easily confuse the model. As a result, it is difficult to distinguish methods which create unnecessary artifacts from those that perform reasonable attributions. To counter this effect, we slightly modify this procedure and propose to remove the k least salient pixels rather than the most salient ones. In this variant, methods that cause the least change in function output better identify unimportant regions in the image. We argue that this benchmark is better as it partially decouples the effects of artifacts from that of removing salient pixels.

Specifically, our procedure is as follows: for a given value of k, we replace the k image pixels corresponding to k least saliency values with black pixels. We measure the neural network function output for the most confident class, before and after perturbation, and plot the absolute value of the fractional difference. We use our pixel perturbation test to evaluate full-gradient saliency maps on the Imagenet 2012 validation dataset, using a VGG-16 model with batch normalization. We compare with gradCAM (Selvaraju et al., 2017), input-gradients (Simonyan et al., 2013), smooth-grad (Smilkov et al., 2017) and integrated gradients (Sundararajan et al., 2017). For this test, we also measure the effect of random pixel removal as a baseline to estimate the effect of artifact creation. We observe that FullGrad causes the least change in output value, and are hence able to better estimate which pixels are unimportant.

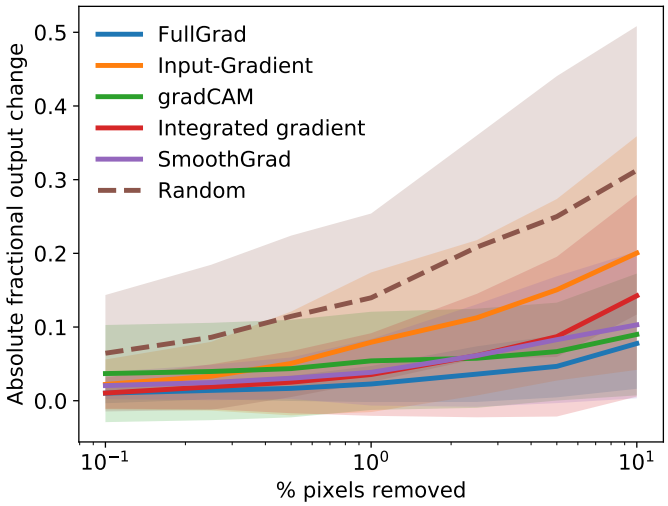
3.5.2 Remove and Retrain

RemOve And Retrain (ROAR) (Hooker et al., 2018) is another approximate benchmark to evaluate how well saliency methods explain model behavior. The test is as follows: *remove* the top-k pixels of an image identified by the saliency map for the entire dataset, and *retrain* a classifier on this modified dataset. If a saliency algorithm indeed correctly identifies the most crucial pixels, then the retrained classifier must have a lower accuracy than the original. Thus an ideal saliency algorithm is one that is able to reduce the accuracy the most upon retraining. Retraining compensates for presence of deletion artifacts caused by removing top-k pixels, which could otherwise mislead the model. This is also not a perfect benchmark, as the retrained model now has additional cues such as the positions of missing pixels, and other visible cues which it had previously ignored. In contrast to the pixel perturbation test which places emphasis on identifying unimportant regions, this test rewards methods that correctly identify important pixels in the image.

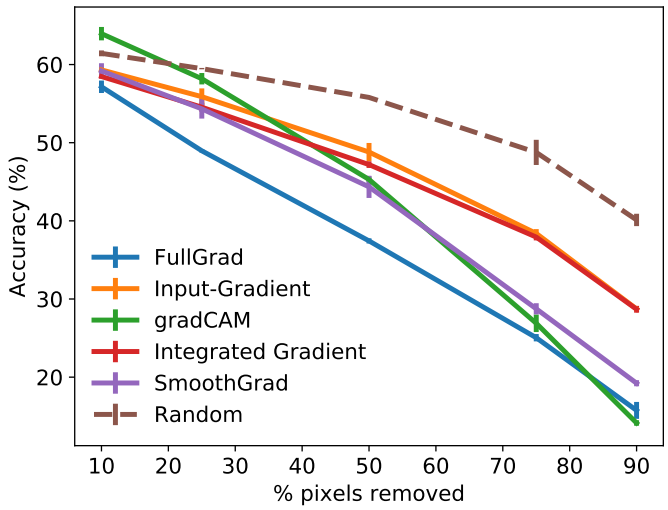
We use ROAR to evaluate full-gradient saliency maps on the CIFAR100 dataset, using a 9-layer VGG model. We compare with gradCAM (Selvaraju et al., 2017), input-gradients (Simonyan et al., 2013), integrated gradients (Sundararajan et al., 2017) and a smooth grad variant called smooth grad squared (Smilkov et al., 2017; Hooker et al., 2018), which was found to perform among the best on this benchmark. We see that FullGrad is indeed able to decrease the accuracy the most when compared to the alternatives, indicating that they correctly identify important pixels in the image.

3.5.3 Visual Inspection

We perform qualitative visual evaluation for FullGrad, along with four baselines: input-gradients (Simonyan et al., 2013), integrated gradients (Sundararajan et al., 2017), smooth grad (Smilkov et al., 2017) and grad-CAM (Selvaraju et al., 2017). We see



(a)



(b)

Figure 3.2 – Quantitative results on saliency map faithfulness. **(a)** Pixel perturbation benchmark (see Section 3.5.1) on Imagenet 2012 validation set where we remove k% least salient pixels and measure absolute value of fractional output change. The **lower** the curve, the better. **(b)** Remove and retrain benchmark (see Section 3.5.2) on CIFAR100 dataset done by removing k% most salient pixels, retraining a classifier and measuring accuracy. The **lower** the accuracy, the better. Results are averaged across three runs. Note that the scales of standard deviation are different for both graphs.

that the first three maps are based on input-gradients alone, and tend to highlight object boundaries more than their interior. Grad-CAM, on the other hand, highlights broad regions of the input without demarcating clear object boundaries. FullGrad combine advantages of both – highlighted regions are confined to object boundaries while highlighting its interior at the same time. This is not surprising as FullGrad includes information both about input-gradients, and also about intermediate-layer gradients like grad-CAM. For input-gradient, integrated gradients and smooth-grad, we do not superimpose the saliency map on the image, as it reduces visual clarity. More comprehensive results without superimposed images for gradCAM and FullGrad are present in the supplementary material.

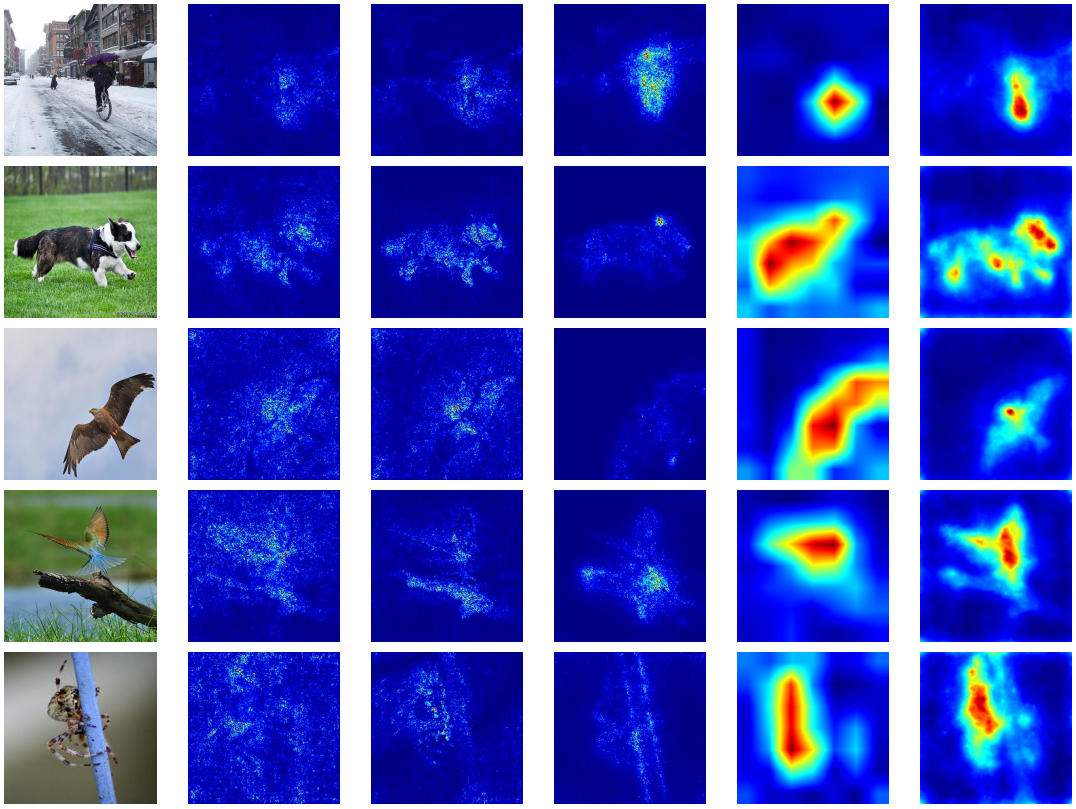


Image	Input gradient (Simonyan et al., 2013)	Integrated gradient (Sundararajan et al., 2017)	Smooth-grad (Smilkov et al., 2017)	Grad-CAM (Selvaraju et al., 2017)	**FullGrad (Ours)**

Figure 3.3 – Comparison of different neural network saliency methods. Integrated-gradients (Sundararajan et al., 2017) and smooth-grad (Smilkov et al., 2017) produce noisy object boundaries, while grad-CAM (Selvaraju et al., 2017) indicates important regions without adhering to boundaries. FullGrad combine both desirable attributes by highlighting salient regions while being tightly confined within objects. For more results, please see the appendix.

3.6 How to Choose $\psi(\cdot)$

In this section, we shall discuss the trade-offs that arise with particular choices of the post-processing function $\psi(\cdot)$, which is central to the reduction from full-gradients to FullGrad. Note that by Proposition 4, any post-processing function cannot satisfy all properties we would like as the resulting representation would still be saliency-based. This implies that any particular choice of post-processing would prioritize satisfying some properties over others.

For example, the post-processing function used in this work is suited to perform well with the commonly used evaluation metrics of pixel perturbation and ROAR for image data. These metrics emphasize highlighting important regions, and thus the magnitude of saliency seems to be more important than the sign. However there are other metrics where this form of post-processing does not perform well. One example is the digit-flipping experiment (Shrikumar et al., 2017), where an example task is to turn images of the MNIST digit "8" into those of the digit "3" by removing pixels which provide positive evidence of "8" and negative evidence for "3". This task emphasizes signed saliency maps, and hence the proposed FullGrad post-processing does not work well here. Having said that, we found that a minimal form of post-processing, with $\psi_m(\cdot) =$ `bilinearUpsample(·)` performed much better on this task. However, this post-processing resulted in a drop in performance on the primary metrics of pixel perturbation and ROAR. Apart from this, we also found that pixel perturbation experiments worked much better on MNIST with $\psi_{mnist}(\cdot) =$ `bilinearUpsample(abs(·))`, which was not the case for Imagenet / CIFAR100. Thus it seems that the post-processing method to use may depend both on the metric and the dataset under consideration. Full details of these experiments are presented in the supplementary material.

We thus provide the following **recommendation to practitioners**: choose the post-processing function based on the evaluation metrics that are most relevant to the application and datasets considered. For most computer vision applications, we believe that the proposed FullGrad post-processing may be sufficient. However, this might not hold for all domains and it might be important to define good evaluation metrics for each case in consultation with domain experts to ascertain the faithfulness of saliency methods to the underlying neural net functions. These issues arise because saliency maps are approximate representations of neural net functionality as shown in Proposition 4, and the numerical quantities in the full-gradient representation (equation 3.3) could be visualized in alternate ways.

3.7 Conclusions and Future Work

In this work, we proposed a novel technique dubbed FullGrad to visualize the function mapping learnt by neural networks. This is done by providing attributions to both the

inputs and the neurons of intermediate layers. Input attributions code for sensitivity to individual input features, while neuron attributions account for interactions between the input features. Individually, they satisfy *weak dependence*, a weak notion for local attribution. Together, they satisfy *completeness*, a desirable property for global attribution.

The inability of saliency methods to satisfy multiple intuitive properties both in theory and practice, has important implications for interpretability. First, it shows that saliency methods are too limiting and that we may need more expressive schemes that allow satisfying multiple such properties simultaneously. Second, it may be the case that all interpretability methods have such trade-offs, in which case we must specify what these trade-offs are in advance for each such method for the benefit of domain experts. Third, it may also be the case that multiple properties might be mathematically irreconcilable, which implies that interpretability may be achievable only in a narrow and specific sense.

Another point of contention with saliency maps is the lack of unambiguous evaluation metrics. This is tautological; if an unambiguous metric indeed existed, the optimal strategy would involve directly optimizing over that metric rather than use saliency maps. One possible avenue for future work may be to define such clear metrics and build models that are trained to satisfy them, thus being interpretable by design.

Appendix

3.8 Proof of Incompatibility

Definition 3. *(Weak dependence on inputs) Consider a piecewise-linear model*

$$f(\mathbf{x}) = \begin{cases} \mathbf{w}_1^T \mathbf{x} + b_1 & \mathbf{x} \in \mathcal{U}_1 \\ ... & \\ \mathbf{w}_n^T \mathbf{x} + b_n & \mathbf{x} \in \mathcal{U}_n \end{cases}$$

where all $\mathcal{U}_i$ are open connected sets. For this function, the saliency map $S(\mathbf{x}) = \sigma(f, \mathbf{x})$ for some $\mathbf{x} \in \mathcal{U}_i$ is constant for all $\mathbf{x} \in \mathcal{U}_i$ and is a function of only the parameters $\mathbf{w}_i, b_i$.

Definition 4. *(Completeness) A saliency map $S(\mathbf{x})$ is*

- *complete if there exists a function ϕ such that $\phi(S(\mathbf{x}), \mathbf{x}) = f(\mathbf{x})$.*
- *complete with a baseline $\mathbf{x}_0$ if there exists a function ϕ_c such that $\phi_c(S(\mathbf{x}), S_0(\mathbf{x}_0), \mathbf{x}, \mathbf{x}_0) = f(\mathbf{x}) - f(\mathbf{x}_0)$, where $S_0(\mathbf{x}_0)$ is the saliency of the baseline.*

Proposition 7. *For any piecewise-linear function f, it is impossible to obtain a saliency map S that satisfies both completeness and weak dependence on inputs, in general*

Proof. From the definition of saliency maps, there exists a mapping from $\sigma : (f, \mathbf{x}) \to S$. Let us consider the family of piecewise linear functions which are defined over the same open connected sets given by $\mathcal{U}_i$ for $i \in [1, n]$. Members of this family thus can be completely specified by the set of parameters $\theta = \{\mathbf{w}_i, b_i | i \in [1, n]\} \in \mathbb{R}^{n*(D+1)}$ for f and similarly θ' for f'.

For this family, **weak dependence** implies that the restriction of the mapping σ to the set $\mathcal{U}_i$, is denoted by $\sigma\Big|_{\mathcal{U}_i} : (\mathbf{w}_i, b_i) \to S$. Now, since $(\mathbf{w}_i, b_i) \in \mathbb{R}^{D+1}$ and $S \in \mathbb{R}^D$, the mapping $\sigma\Big|_{\mathcal{U}_i}$ is a many-to-one function. This implies that there exists piecewise linear functions f and f' within this family, with parameters $\theta_i = (\mathbf{w}_i, b_i)$ and $\theta_i' = (\mathbf{w}'_i, b'_i)$ respectively (with $\theta_i \neq \theta_i'$), which map to the same saliency map S.

Part (a): From the first definition of **completeness**, there exists a mapping $\phi : S, \mathbf{x} \rightarrow f(\mathbf{x})$. However, for two different piecewise linear functions f and f' that map to the same S for some input $\mathbf{x} \in \mathcal{U}_i$, we must have that $\phi(S, \mathbf{x}) = f(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + b_i$ for f and $\phi(S, \mathbf{x}) = f'(\mathbf{x}) = \mathbf{w'}_i^T\mathbf{x} + b'_i$ for f'. This can hold for a local neighbourhood around $\mathbf{x}$ if and only if $\mathbf{w}_i = \mathbf{w'}_i$ and $b_i = b'_i$, which we have already assumed to be not true.

Part (b): From the second definition of **completeness**, there exists a mapping $\phi_c : S, S_0, \mathbf{x}, \mathbf{x}_0 \rightarrow f(\mathbf{x}) - f(\mathbf{x}_0)$.

Let the baseline input $\mathbf{x}_0 \in \mathcal{U}_j$. Similar to the case above, let us assume existence of functions f and f' with parameters $\theta_j = (\mathbf{w}_j, b_j)$ and $\theta'_j = (\mathbf{w'}_j, b'_j)$ respectively (with $\theta_j \neq \theta'_j$), which map to the same saliency map S_0. This condition is in addition to the condition already applied on $\mathcal{U}_i$.

Hence we must have that $\phi(S, S_0, \mathbf{x}, \mathbf{x}_0) = f(\mathbf{x}) - f(\mathbf{x}_0) = \mathbf{w}_i^T\mathbf{x} + b_i - \mathbf{w}_j^T\mathbf{x} - b_j$ for f and $\phi(S, S_0, \mathbf{x}, \mathbf{x}_0) = f'(\mathbf{x}) - f'(\mathbf{x}_0) = \mathbf{w'}_i^T\mathbf{x} + b'_i - \mathbf{w'}_j^T\mathbf{x} - b'_j$ for f'. This can hold for local neighbourhoods around $\mathbf{x}$ and $\mathbf{x}_0$ if and only if $\mathbf{w}_i = \mathbf{w'}_i$, $\mathbf{w}_j = \mathbf{w'}_j$ and $b_i - b'_i = b_j - b'_j$. The final condition does not hold in general, hence completeness is not satisfied. □

When does $b_i - b'_i = b_j - b'_j$ hold?

- For piecewise linear models without bias terms (e.g.: ReLU neural networks with no biases), the terms b_i, b'_i, b_j, b'_j are all zero, and hence this condition holds for such networks.
- For linear models, (as opposed to piecewise linear models), or when both $\mathbf{x}$ and $\mathbf{x}_0$ lie on the same linear piece, then $b_i = b_j$, which automatically implies that the condition holds.

However these are corner cases and the condition on the biases does not hold in general.

3.9 Full-gradient Proofs

Proposition 8. *Let f be a ReLU neural network without bias units, then $f(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x})^T\mathbf{x}$.*

Proof. For ReLU nets without bias, we have $f(k\mathbf{x}) = kf(\mathbf{x})$ for $k \geq 0$. This is a consequence of the positive homogeneity property of ReLU (i.e; $\max(0, k\mathbf{x}) = k\max(0, \mathbf{x})$)

Now let $\epsilon \in \mathbb{R}^+$ be infinitesimally small. We can now use first-order Taylor series to write the following. $f((1+\epsilon)\mathbf{x}) = f(\mathbf{x}) + \epsilon f(\mathbf{x}) = f(\mathbf{x}) + \epsilon\mathbf{x}^T\nabla_{\mathbf{x}} f(\mathbf{x})$. □

Proposition 9. *Let f be a ReLU neural network with bias-parameters $\mathbf{b} \in \mathbb{R}^F$, then*

$$\begin{aligned} f(\mathbf{x};\mathbf{b}) &= \nabla_{\mathbf{x}} f(\mathbf{x};\mathbf{b})^T \mathbf{x} + \sum_{i\in[1,F]} (\nabla_b f(\mathbf{x};\mathbf{b}) \odot \mathbf{b})_i \\ &= \nabla_{\mathbf{x}} f(\mathbf{x};\mathbf{b})^T \mathbf{x} + \nabla_b f(\mathbf{x};\mathbf{b})^T \mathbf{b} \end{aligned} \tag{3.3}$$

Proof. We introduce bias inputs $\mathbf{x}_b = \mathbf{1}^F$, an all-ones vector, which are multiplied with bias-parameters $\mathbf{b}$. Now $f(\mathbf{x}, \mathbf{x}_b)$ is a linear function with inputs $(\mathbf{x}, \mathbf{x}_b)$. Proposition applies here.

$$\begin{aligned} f(\mathbf{x}, \mathbf{x}_b) &= \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{x}_b)^T \mathbf{x} + \nabla_{\mathbf{x}_b} f(\mathbf{x}, \mathbf{x}_b)^T \mathbf{x}_b \\ &= \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{x}_b)^T \mathbf{x} + \sum_i (\nabla_{\mathbf{x}_b} f(\mathbf{x}, \mathbf{x}_b))_i \end{aligned} \tag{3.4}$$

Using chain rule for ReLU networks, we have $\nabla_{\mathbf{x}_b} f(\mathbf{x}, \mathbf{x}_b; \mathbf{b}, \mathbf{z}) = \nabla_z f(\mathbf{x}, \mathbf{x}_b; \mathbf{b}, \mathbf{z}) \odot \mathbf{b}$, where $\mathbf{z} \in R^F$ consists of all intermediate pre-activations. Again invoking chain rule, we have $\nabla_z f(\mathbf{x}, \mathbf{x}_b; \mathbf{b}, \mathbf{z}) = \nabla_b f(\mathbf{x}, \mathbf{x}_b; \mathbf{b}, \mathbf{z})$

□

Observation. *For a piecewise linear neural network, $f^b(\mathbf{x})$ is locally constant in each linear region.*

Proof. Consider a one-hidden layer ReLU net of the form $f(x) = w_1 * \text{relu}(w_0 * x + b_0) + b_1$, where $f(\mathbf{x}) \in \mathbb{R}$. Let $\rho(z) = \frac{d\text{relu}(z)}{dz}$ be the derivative of the output of relu w.r.t. its inputs. Then the gradients w.r.t. b_0 can be written as $\frac{df}{db_0} = w_1 * \rho(w_0 * x + b_0)$. For each linear region, the derivatives of the relu non-linearities w.r.t. their inputs are constant. Thus for a one-hidden layer net, the bias-gradients are constant in each linear region. The same can be recursively applied for deeper networks. □

3.10 Experiments to Illustrate Post-Processing Trade-offs

In this section, we shall describe the experiments performed on the MNIST dataset. First, we perform the digit flipping experiment Shrikumar et al. (2017) to test class sensitivity of our method. Next, we perform pixel perturbation experiment as outlined in Section 5.1 of the main chapter.

Method	Random	Gradient	IntegratedGrad	FullGrad	FullGrad (no abs)
Δ log-odds	1.41 ± 8.21	11.92 ± 17.99	10.81 ± 20.11	8.26 ± 21.44	$\mathbf{12.93 \pm 18.20}$

Table 3.1 – Results on the digit flipping task ($8 \rightarrow 3$). We see that FullGrad (minimal) outperforms others including FullGrad. Larger numbers are better.

3.10.1 Digit Flipping

Broadly, the task here is to turn images of the MNIST digit "8" into those of the digit "3" by removing pixels which provide positive evidence of "8" and negative evidence for "3". We perform experiments with a setting similar to the DeepLIFT paper Shrikumar et al. (2017), except that we use a VGG-like architecture. Here, FullGrad (no abs) refers to using $\psi_m(\cdot) =$ `bilinearUpsample(·)` and the FullGrad method refers to using $\psi_m(\cdot) =$ `bilinearUpsample(abs(·))`. From the results in Table 3.1, we see that FullGrad without absolute value performs better in the digit flipping task when compared to FullGrad and all other methods.

3.10.2 Pixel Perturbation

We perform the pixel perturbation task on MNIST. This involves removing the least salient pixels as predicted by a saliency map method and measuring the fractional change in output. The smaller the fractional output change, the better is the saliency method. From Table 3.2, we observe that Integrated gradients perform best overall for this dataset. We hypothesize that the binary nature of MNIST data (i.e.; pixels are either black or white, and "removed" pixels are black) may be well-suited to Integrated gradients, which is not the case for our Imagenet experiments. However, more interestingly, we observe that regular FullGrad outperforms the variant without absolute values.

Thus while for digit flipping it seems that FullGrad (no abs) is the best, followed by gradients and Integrated gradients, for pixel perturbation it seems that Integrated Gradients is the best followed by FullGrad and FullGrad (no abs). Thus it seems that any single saliency or post-processing method is never consistently better than the others, which might point to either the deficiency of the methods themselves, or the complementary nature of the metrics.

3.11 Saliency Results

Method	Random	Gradient	IntegratedGrad	FullGrad	FullGrad (no abs)
RF = 0.5	0.82 ± 0.28	0.29 ± 0.22	$\mathbf{0 \pm 0}$	0.06 ± 0.13	0.19 ± 0.19
RF = 0.7	0.98 ± 0.34	0.52 ± 0.27	$\mathbf{0.004 \pm 0.06}$	0.08 ± 0.11	0.34 ± 0.23
RF = 0.9	1.12 ± 0.42	0.88 ± 0.34	$\mathbf{0.44 \pm 0.33}$	0.55 ± 0.30	0.63 ± 0.27

Table 3.2 – Results on the pixel perturbation task on MNIST. In this case, FullGrad performs better than FullGrad (minimal). The overall best performer here is Integrated gradients. Smaller numbers are better.

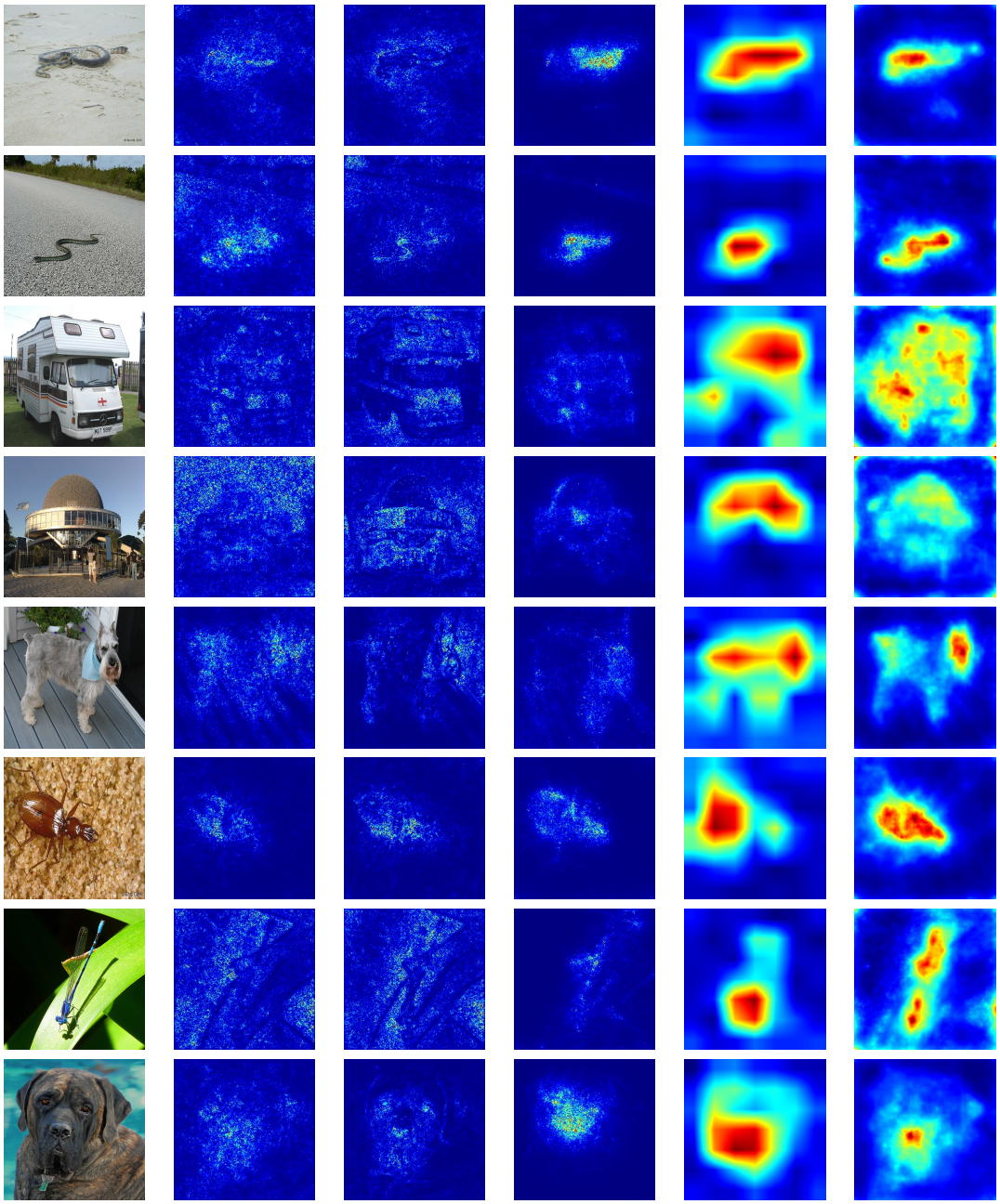


Image	Input gradient Simonyan et al. (2013)	Integrated gradient Sundararajan et al. (2017)	Smooth-grad Smilkov et al. (2017)	Grad-CAM Selvaraju et al. (2017)	**FullGrad (Ours)**

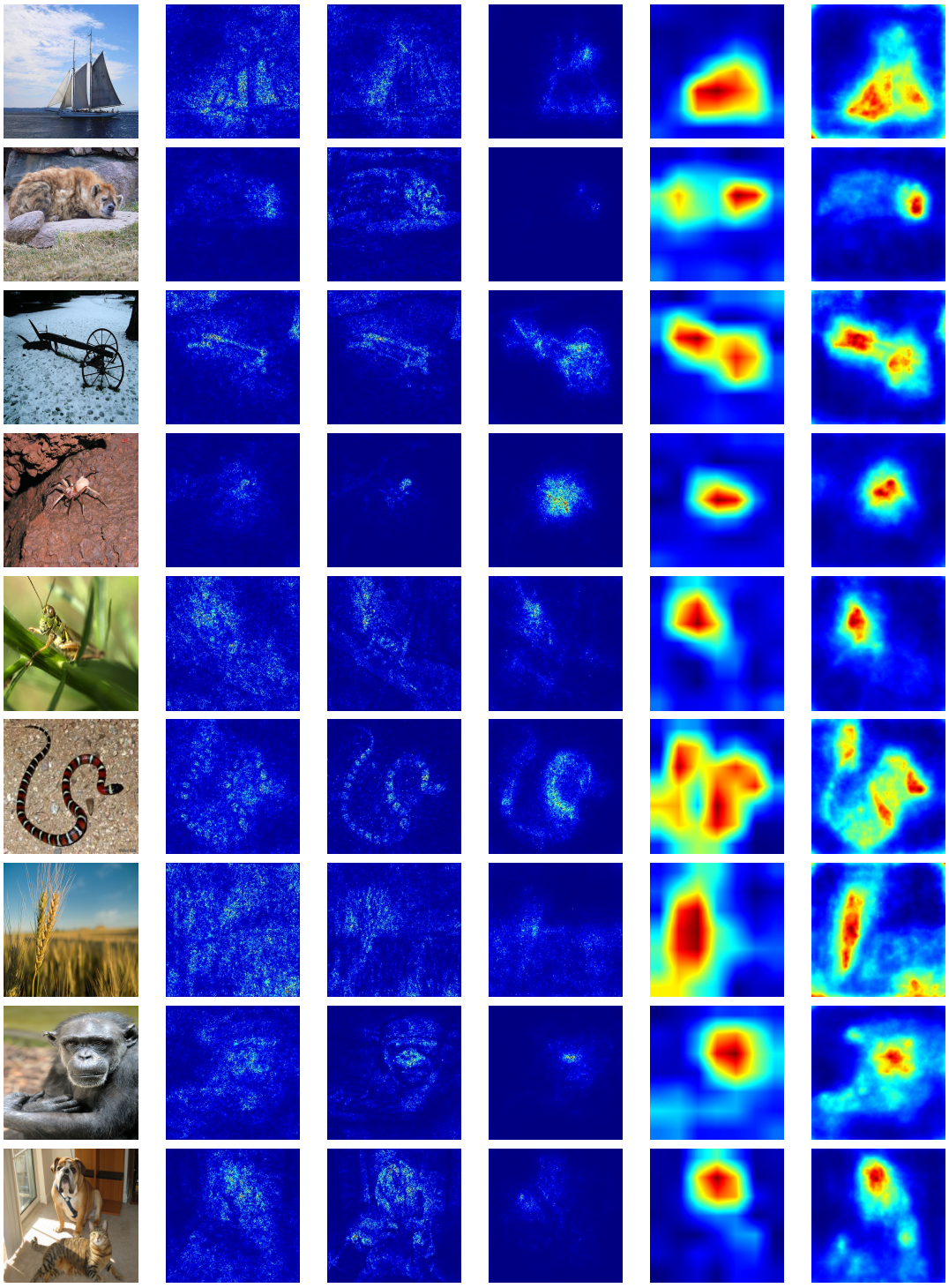


Image | Input gradient Simonyan et al. (2013) | Integrated gradient Sundararajan et al. (2017) | Smooth-grad Smilkov et al. (2017) | Grad-CAM Selvaraju et al. (2017) | **FullGrad (Ours)**

Figure 3.4 – Comparison of different neural network saliency methods.

4 Knowledge Transfer with Full-Gradient Matching

In this chapter, we use the full-gradient representation for the task of distillation and model regularization. We first use these full-gradients for distillation by aligning gradients of intermediate representations of a teacher and a student model. Next, we regularize bias-gradients alone to improve generalization, connecting it with noise injection methods like dropout. Experimental results show improved distillation on small data-sets and improved generalization for neural network training.

4.1 Introduction

One important problem in machine learning is to optimally incorporate prior knowledge about data into models. Such prior knowledge informs regularization methods, which help improve generalization when dealing with small training sets. This is especially crucial for knowledge transfer, which involves emulating the function mapping of a "teacher" in a "student" using training examples. For this task, prior knowledge encodes information about the teacher's map. A good representation of this map can result in rapid learning by the student using little data.

In Chapter 2 we proposed to use input-gradients, i.e., the gradients of the outputs w.r.t. input, for knowledge transfer. Input-gradients capture the slope of the local affine approximation of the neural network. Together with the function output, this method completely captures the local functional behavior of neural nets. While these methods capture functional behaviour, we are often faced with distillation problems where both student and teacher share some common structure, say, both being convolutional models. In such cases, pure functional matching fails to leverage such structural similarity for more efficient distillation. In this chapter, we propose the use of full-gradients, which comprises of both input-gradients and layerwise feature-gradients, and thus capture both properties of the input-output map as well as layerwise maps. We provide experimental evidence showing that full-gradients indeed help knowledge transfer, and

that bias-gradient-norm minimization provides regularization benefits.

4.2 Full-Gradient Matching

Given two networks f and g, we would like to perform distillation with g being the teacher and f being the student. The problem of distillation is to improve the training of f using information from g. In essence, we look for a function f with the same input-output mapping as g but with a different parameterization owing to its different architecture. If the architecture of both models are the same, and if they both have the same full-gradient representation, they necessarily represent the same function. Note that the converse is not true. Leveraging this fact, we propose to match the full-gradient representation of models that have nearly identical architectures in order to enable them to represent the same function.

Assume that both f and g are at least L-layer neural networks, which are not necessarily of the same depth. For example, f can have L layers and g can have $2L$ layers. In such cases, one can divide g into L *sections* such that each section contains 2 layers. Given two such L-section models, one can match the full-gradient representation between them. Given a model f, the full-gradient representation is given below.

$$f(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x})^T \mathbf{x} + \sum_{l=1}^{L} \nabla_{\mathbf{b}_l^f} f(\mathbf{x})^T \mathbf{b}_l^f$$

Here, $\mathbf{b}_l^f$ refers to layer-wise biases for model f. For two such pairs of models we propose to match the bias-gradients of each section of the models separately. Specifically, we would like to ensure that $\nabla_{\mathbf{b}_l^f} f(\mathbf{x}) \approx \nabla_{\mathbf{b}_l^g} g(\mathbf{x})$. However, the bias feature dimensions may not necessarily be the same between two models. To account for this, we propose to match aggregated bias-gradients between two models by summing them channel-wise for convolutional layers. To ensure that the spatial sizes of the bias gradients are the same, we apply a spatial resizing operator to the smaller bias gradient map of the two. Thus the distillation loss function is of the following form.

$$\ell(f, g) = \sum_{l} \|\psi(\nabla_{\mathbf{b}_l^f} f(\mathbf{x})) - \psi(\nabla_{\mathbf{b}_l^g} g(\mathbf{x}))\|^2$$

Here, $\psi(\cdot)$ is a post-processing operator which here is `normalize(sum_channels(resize(·)))`. Here the normalization refers to rescaling the vector such that the norm is unity. Thus in effect here we align the angle of vectors on a unit ball.

We here make of use of the fact that both student and teacher models are multi-layer models. However, does this always hold? Multiple theoretical results about deep networks express so-called "no-flattening" theorems (Cohen et al., 2016; Raghu et al., 2017). Broadly speaking, they state that a shallow network requires exponentially many units to approximate a deep network. In practice for distillation this means that different layers in a neural network are indeed useful and cannot be approximated by shallower nets. Furthermore, visualization studies in computer vision have pointed to the fact that different layers in deep networks have clearly delineated tasks (Zeiler and Fergus, 2014). For instance, early layers often perform edge detection, while higher layers perform object part detection. This implies that structure in the form of depth, and thus the usage of such multi-layer models especially for vision tasks may be unavoidable.

4.3 Bias-Gradient Regularization

In Chapter 2 we interpret input-gradients as the sensitivity of the neural network to noise added to its inputs. Similarly bias-gradients can be interpreted as sensitivity to bias-parameters. This suggests a natural regularization strategy, that of minimizing such sensitivity. Minimizing the sensitivity of a neural network to its parameters has long been considered (Hochreiter and Schmidhuber, 1997) as an important criterion for generalization. Recent works also connect the notion of *flat minimum* to implicit regularization of SGD, thus partially explaining the success of deep learning (Keskar et al., 2016).

Given a neural network function f with weights $\mathbf{w}$ and biases $\mathbf{b} \in \mathbb{R}^F$, we apply multiplicative noise to the biases to obtain the following.

Proposition 10. *Given the notations above, and assuming $y \in \mathbb{R}$, with noise variable $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{1}, \sigma^2\mathcal{I}) \in \mathbb{R}^F$, we have*

$$\mathbb{E}_{\boldsymbol{\xi}}\left[\left(y - f(\mathbf{x}; \mathbf{w}, \mathbf{b} \odot \boldsymbol{\xi})\right)^2\right] \sim \left(y - f(\mathbf{x}; \mathbf{w}, \mathbf{b})\right)^2 + \sigma^2 \|\mathbf{b} \odot \nabla_{\mathbf{b}} f(\mathbf{x}; \mathbf{w}, \mathbf{b})\|_2^2$$

This is obtained from applying first order Taylor series expansion at a local linear neighbourhood around $\mathbf{b}$. This general expression holds for any variable of f. Notice that the second term contains $\mathbf{b} \odot \nabla_{\mathbf{b}} f(\mathbf{x}; \mathbf{w}, \mathbf{b})$, which is exactly the bias-gradient. Hence the bias-gradient can be interpreted as the sensitivity of the neural network to *multiplicative* noise applied to bias-parameters.

One other important regularizer which adds noise to intermediate layers of networks is dropout (Srivastava et al., 2014). However the difference is that while dropout can be viewed as adding multiplicative noise to activations directly, bias-gradient regularization adds multiplicative noise to bias-parameters. Equivalently, this can also be thought of as adding noise to pre-activations of layers, as opposed to post-non-linearity activations

as done typically in dropout.

Let us consider a form of dropout under the limit of low-dropout noise. For convenience we shall assume dropout with multiplicative gaussian noise, but same can be easily repeated with bernoulli noise. Invoking Proposition 10, and using it for an intermediate activation $\mathbf{z} \in \mathbb{R}^m$, we have

$$\mathbb{E}_{\boldsymbol{\xi}} \left[y - f(\mathbf{x}; \mathbf{z} \odot \boldsymbol{\xi}) \right]^2 \sim (y - f(\mathbf{x}; \mathbf{z}))^2 + \sigma^2 \| \mathbf{z} \odot \nabla_{\mathbf{z}} f(\mathbf{x}; \mathbf{z}) \|_2^2 \tag{4.1}$$

Here, $\boldsymbol{\xi} \in \mathbb{R}^m$ is the multiplicative gaussian noise variable. Thus under the low-noise limit, we can analytically perform dropout by taking expectation over all noise terms. This results in a deterministic regularizer which minimizes norm of $\mathbf{z} \odot \nabla_{\mathbf{z}} f(\mathbf{x}; \mathbf{z})$. We observe that this term is similar to bias-gradients as the gradient w.r.t. biases of a layer $\mathbf{b}$ is the same as the gradient w.r.t. the corresponding intermediate pre-activation $\mathbf{z}$, by chain rule. Note that both regularizers are identical when the previous layer's activations are zero, thus making $\mathbf{z} = \mathbf{b}$. To summarize, dropout and bias-gradient regularization share a tight connection, that of reducing the sensitivity of the output to multiplicative noise added at the intermediate layers.

4.4 Experiments

To show the effectiveness of full-gradients, we run experiments on distillation, regularization and visualization. First, we perform distillation on CIFAR-100 datasets (Krizhevsky and Hinton, 2009) in a limited-data setting. Second, we regularize training of individual neural networks on the CIFAR100 dataset. Finally, we show visualizations of neural network saliency maps using full-gradient visualization. For all experiments, we approximate gradient computation by computing gradient of the output unit with the correct class, as done by Srinivas and Fleuret (2018). Details about experiments are present in the supplementary material.

4.4.1 Distillation

For distillation experiments, we use VGG-like (Simonyan and Zisserman, 2014) architectures with batch normalization. The main difference is we discard all fully-connected layers except the final. We use the following procedure in our experiments. First, a 9-layer "teacher" network is trained on the full CIFAR-100 dataset. Then, a larger 13-layer "student" network is trained, but this time on small subsets rather than the full dataset. As the teacher is trained on much more data than the student, we expect distillation to improve the student's performance. Note that in this case our objective is

not to compress the teacher model, but to effectively transfer the knowledge of the full CIFAR-100 dataset when only limited samples are available.

We compare our methods against the following baselines.

1. **Cross-Entropy (CE) training** – Here we train the student using only the ground truth (hard labels) available with the dataset without invoking the teacher network.

2. **CE + match output-activations (Activation Matching)** – This is the classical form of distillation (Ba and Caruana, 2014; Hinton et al., 2015), where the output-activations of the teacher network are matched with that of the student. This is weighted with the cross-entropy term which uses ground truth targets. Here we use the squared-error loss function for matching activations.

3. **CE + match {output-activations + input-gradients } (i-gradients)** – This is the regularizer used by (Czarnecki et al., 2017; Srinivas and Fleuret, 2018), where the input-gradients of teacher and student networks are matched. Here we minimize the ℓ_2 distance between input-gradients.

4. **CE + match { output-activations + hidden-layer-attention} (Attention)** – This approach is taken by Zagoruyko and Komodakis (2017), who match the channel-wise absolute sum of hidden layers for teacher and student with layers of same spatial dimensions. This can also be thought of as matching intermediate activations rather than intermediate gradients like our method does.

5. **i-gradients + Attention** – Considering that attention mapping also incorporates sub-structure information like bias-gradients, we combine two previous baselines to directly compare against our method.

We find that our new augmented baseline of input-gradients with attention matching is surprisingly strong and beats all previous baselines, including full-gradients. To improve upon this strong baseline, we add to it the bias-gradient matching term and find that it improves performance over that. This seems to contradict our assertions in section 4.1 that one can match either bias-gradients or intermediate activations to account for sub-structure, as they contain information about the same affine plane.

However individually, these quantities carry complementary information. While attention maps at a layer capture computation performed by the neural network upto that layer, the gradients from outputs w.r.t. a layer capture the computation done by the rest of the network after that layer. We match bias-gradients or attention maps of only three convolutional layers out of eleven. This is done because computing these for all layers during training is computationally expensive. This explains the increase in performance for this augmented objective. Similar experiments are presented for CIFAR-10.

Table 4.1 – Distillation performance on CIFAR100 (see Section 4.4.1). Table shows average test accuracy (%) across two runs, along with standard deviation. We find that matching Full-gradients along with attention works best for limited-data settings. The student network is VGG-11 while the teacher is a VGG-9 network which achieves 66.82% accuracy. As the student is larger than the teacher, distillation does not help when using the entire dataset.

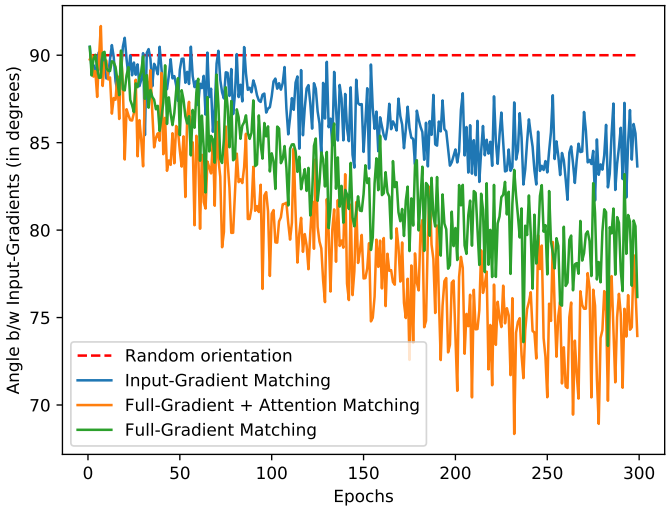
# data / class →	5	10	50	100	500 (full)
Cross-Entropy training	7.45 ± 0.3	11.83 ± 0.4	40.88 ± 0.8	51.19 ± 0.01	**69.95** ± 0.2
Match Activations	23.72 ± 1.3	37.22 ± 0.2	59.43 ± 0.02	63.91 ± 0.2	66.99 ± 0.2
Match i-gradients (iG)	27.27 ± 1.2	41.47 ± 1	61.83 ± 0.01	65.43 ± 0.6	66.92 ± 0.7
Match Attention	38.18 ± 1.9	46.39 ± 0.1	60.27 ± 0.3	64.28 ± 0.2	66.53 ± 0.3
Match { iG + Att. }	42.75 ± 1.7	51.16 ± 0.6	**62.62** ± 0.6	**65.38** ± 0.2	67.25 ± 0.8
Match Fullgrad (FG)	35.15 ± 0.5	48.00 ± 0.4	**62.88** ± 0.1	**65.84** ± 0.1	66.83 ± 0.1
Match { FG + Att. }	**47.11** ± 0.9	**54.59** ± 0.2	**63.20** ± 0.4	**65.49** ± 0.1	66.65 ± 0.4

Effect on Input-gradient Matching

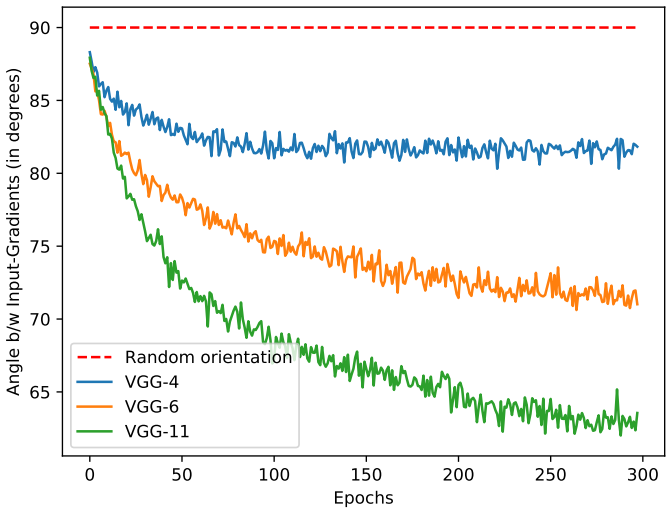
In our experiments we found that the gradient-based matching terms are difficult to optimize. This was also observed by (Srinivas and Fleuret, 2018), who attributed this to a second-order vanishing-gradient effect. We did not observe any such effect in our experiments, and we are unsure of the exact cause of this difficulty. Figure 4.1a illustrates this phenomenon for CIFAR100 distillation with 5 data points per class. For the case of input-gradient matching, we see that the cosine angle hardly drops below 85° on the training set. Surprisingly, augmenting this loss with bias-gradient or attention losses helps the optimization of input-gradients. In all three cases, the regularization constant for input-gradient matching loss term is unchanged. This indicates that the gains we observe could be because of this virtuous cycle of regularizers reinforcing and improving each others' objectives.

Effect of Student size

Common folk wisdom among machine learning researchers is that small models must be preferred to large ones when training with limited data. We find that this advice does not hold for the case of distillation. We train three models (VGG-{4,6,11}) on CIFAR100 with 50 data points per class with full-gradient matching. We find that surprisingly, the larger models perform better. For VGG-11, we get an accuracy of 62.95%, while for VGG-6 and VGG-4 we get 58.08% and 50.87% respectively. We also plot the angle between input-gradients for all three cases in figure 4.1b, and find that the input-gradient norms are better aligned for VGG-11. These observations are not surprising, as additional capacity is required to fit all the objectives we introduce.



(a)



(b)

Figure 4.1 – Plot **(a)** shows the evolution of input-gradient angle between teacher and student during training. The input-gradient matching objective is identical in all three cases, and we find that augmenting this with full-gradient and attention matching helps increase alignment. Plot **(b)** shows the evolution of input-gradient angle between teacher and student for three different student networks. We find that larger models fit the teacher better, which is also reflected in the improved input-gradient alignment.

Table 4.2 – Distillation performance on CIFAR10 (see Section 4.4.1). Table shows average test accuracy (%) across two runs, along with standard deviation. We find that matching Full-gradients along with attention works best for limited-data settings. The student network is VGG-11 while the teacher is a VGG-9 network which achieves 90.49% accuracy. As the student is larger than the teacher, distillation does not help when using the entire dataset

# Data / class →	50	100	500	1000	5000
Cross-Entropy training	49.29 ± 1.6	59.93± 0.1	79.36 ± 0.04	83.87 ± 0.1	91.95 ± 0.1
Match Activations	55.43 ± 2.1	65.33 ± 2.2	85.44 ± 0.1	88.77 ± 0.3	**92.47** ± 0.1
Match i-gradients (iG)	55.73 ± 2	67.22 ± 3.0	85.84 ± 0.1	89.30 ± 0.3	92.04 ± 0.01
Match Attention	68.11 ± 0.8	74.44 ± 0.2	85.88 ± 0.1	88.61 ± 0.1	91.20 ± 0.01
Match {iG + Att.}	70.83 ± 1.0	77.06 ± 0.2	86.51 ± 0.3	**89.63** ± 0.1	90.68 ± 0.04
Match Fullgrad (FG)	58.88 ± 0.2	69.42 ± 1.4	86.55 ± 0.1	**89.76** ± 0.1	91.49 ± 0.05
Match {FG + Att.}	**72.75** ± 0.4	**78.71** ± 0.1	**87.31** ± 0.3	**89.87** ± 0.3	90.68 ± 0.1

We make two additional observations here. First, when using VGG-9 as student, we found that it performed as good as VGG-11. This is expected as the teacher itself is a VGG-9 network. Second, VGG-4 and 6 do slightly outperform VGG-11 on smaller datasets such as using 5 points per class, and show better input-gradient alignment. However we did not observe this for other cases.

Table 4.3 – Regularization of VGG-11 models on CIFAR100 (see Section 4.4.2). We report average test accuracy (%) across two runs, along with standard deviation. λs denote regularization strengths, while p is dropout probability. We apply these to the same single layer of VGG-11, and find that bias-gradient regularization outperforms dropout and bias weight decay in all cases.

# Data points / class →	50	100	500
No regularization	33.25 ± 0.6	46.24 ± 0.1	68.48 ± 0.1
Dropout (Srivastava et al., 2014)	35.04 ± 0.6	47.62 ± 0.7	70.14 ± 0.06
Bias weight decay	34.17 ± 0.2	47.29 ± 0.7	68.75 ± 0.04
Bias-gradient (Ours)	**36.02** ± 0.08	**48.76** ± 0.1	**71.49** ± 0.02

4.4.2 Regularization

We perform experiments where we penalize the bias-gradient norm to check whether it improves generalization. We train 9-layer VGG networks on CIFAR100 with varying number of data points per class, and measure test accuracy. We compare our method with dropout and bias parameter weight-decay applied to the same layer whose bias-gradient norm we compute. We also found that regularization benefits arise when applying these regularizers to final convolutional layers. For all methods, we choose reg-

ularization constants by performing grid search, leading to using $p = 0.5$ for dropout, $\lambda = 1e-1$ for bias-weight decay, and $\lambda = 1e2$ for bias-gradient regularization.

Our experiments confirm our hypothesis that bias-gradients have regularization benefits, and we find that they are also superior to dropout and weight decay on biases.

4.5 Conclusion

We have introduced the full-gradient representation, which completely captures the local affine behaviour of a neural network. In particular, it provides a formal way to reason about the intermediate layers of multi-layered architectures. In this chapter, we used this representation to perform distillation and regularization which drew parallels with dropout.

Despite these advances, this representation is incomplete without a formal understanding of structural similarities between neural nets. This was briefly discussed in Section 4.1. Future work can focus on formalizing this notion for convolutional networks, as well as on methods to automatically discover such similarity between two architectures and find the optimal matching losses for knowledge transfer.

Appendix

4.6 Proofs

Proposition 11. *Given the notations above, and assuming $y \in \mathbb{R}$, with noise variable $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{1}, \sigma^2\mathcal{I}) \in \mathbb{R}^f$, we have*

$$\mathbb{E}_{\boldsymbol{\xi}}\left[(y - f(\mathbf{x}; \mathbf{w}, \mathbf{b} \odot \boldsymbol{\xi}))^2\right] \sim (y - f(\mathbf{x}; \mathbf{w}, \mathbf{b}))^2 + \sigma^2 \|\mathbf{b} \odot \nabla_{\mathbf{b}} f(\mathbf{x}; \mathbf{w}, \mathbf{b})\|_2^2$$

Proof. There exists σ and $\boldsymbol{\xi} \sim \mathcal{N}(1, \sigma^2)$ small enough that first-order Taylor series expansion holds true. We first split $\mathbf{b} \odot \boldsymbol{\xi} = \mathbf{b} + \mathbf{b} \odot (\boldsymbol{\xi} - 1)$. Notice $\boldsymbol{\xi} - 1 \sim \mathcal{N}(0, \sigma^2)$. Let $\mathbf{b} \odot \boldsymbol{\xi} - 1 = \phi \sim \mathcal{N}(0, \mathbf{b}\mathcal{I}\sigma^2)$

$$\begin{aligned} &\mathbb{E}_{\phi}\left[y - f(\mathbf{x}, \mathbf{b} + \phi)\right]^2 \\ &\sim \left[y - f(\mathbf{x}, \mathbf{b})\right]^2 + \mathbb{E}_{\phi}\left[\phi^T \nabla_b f(\mathbf{x}, \mathbf{b})\right]^2 \qquad (4.2) \\ &= \left[y - f(\mathbf{x}, \mathbf{b})\right]^2 + \mathbb{E}_{\phi}[{\phi^2}^T \nabla_b f(\mathbf{x}, \mathbf{b})^2] \\ &= \left[y - f(\mathbf{x}, \mathbf{b})\right]^2 + \sigma^2 \|\mathbf{b} \odot \nabla_b f(\mathbf{x}, \mathbf{b})\|^2 \end{aligned}$$

Equation 4.2 follows from applying zero mean assumption on ϕ. Then we apply the diagonal covariance assumption, after which we simply evaluate the expectation.

□

4.7 Experimental details

4.7.1 Network Architectures

The architecture for our networks follow the VGG design philosophy. Specifically, we have blocks with the following elements:

- 3×3 conv kernels with c channels of stride 1
- Batch Normalization
- ReLU

Whenever we use Max-pooling (M), we use stride 2 and window size 2.

The architecture for VGG-9 is - $[64 - M - 128 - M - 256 - 256 - M - 512 - 512 - M - 512 - 512 - M]$. Here, the number stands for the number of convolution channels, and M represents max-pooling. At the end of all the convolutional and max-pooling layers, we have a Global Average Pooling (GAP) layer, after which we have a fully connected layer leading up to the final classes. Similar architecture is used for both CIFAR-10 and CIFAR-100 experiments.

4.7.2 Loss function

The loss function for distillation experiments use the following form.

$$\begin{aligned} \ell(f(\mathbf{x}), g(|X)) &= \alpha \times (\text{CE}) + \beta \times (\text{Match Activations}) \\ &+ \gamma \times (\text{Match inputgradients}) \\ &+ \delta \times (\text{Match biasgradients}) \end{aligned}$$

In our experiments, $\alpha, \beta, \gamma, \delta$ are either set to 10 or 0. In other words, all regularization constants are 10.

Here, 'CE' refers to cross-entropy with ground truth labels. 'Match Activations' refers to squared error term over pre-softmax activations of the form $(y_s - y_t)^2$. 'Match inputgradients' refers to the same squared error term, but for gradients. For matching bias-gradients, we choose three layers at three different spatial resolutions for student and teacher. These layers had $64, 128, 256$ channels each. We found matching early layers to be more beneficial in general. The loss function used for matching bias-gradients is -

$$\text{Match gradients} = \left|\left| \frac{f^b(\mathbf{x})}{\|f^b(\mathbf{x})\|_2} - \frac{g^b(\mathbf{x})}{\|g^b(\mathbf{x})\|_2} \right|\right|_2^2 \tag{4.3}$$

For notational convenience, $f^b(\mathbf{x})$ here refers to the channel-summed bias-gradient of a layer rather than the full bias-gradient.

Optimization

For CIFAR-100 distillation experiments, we run optimization for 300 epochs. We use the Adam optimizer, with an initial learning rate of $1e-3$, and a single learning rate annealing (to $1e-4$) at 250 epochs. We used a batch size of 128. We use similar parameters for CIFAR-10. For regularization experiments, we ran optimization for 100 epochs, with annealing at 80 epochs.

5 Rethinking the Role of Gradient-based Saliency Methods

Current methods for the interpretability of discriminative deep neural networks commonly rely on the model's input-gradients, i.e., the gradients of the output logits w.r.t. the inputs. The common assumption is that these input-gradients contain information regarding $p_\theta(y \mid \mathbf{x})$, the model's discriminative capabilities, thus justifying their use for interpretability. However, in this work we show that these input-gradients can be arbitrarily manipulated as a consequence of the shift-invariance of softmax without changing the discriminative function. This leaves an open question: if input-gradients can be arbitrary, why are they highly structured and explanatory in standard models?

We investigate this by re-interpreting the logits of standard softmax-based classifiers as unnormalized log-densities of the data distribution and show that input-gradients can be viewed as gradients of a class-conditional density model $p_\theta(\mathbf{x} \mid y)$ implicit within the discriminative model. This leads us to hypothesize that the highly structured and explanatory nature of input-gradients may be due to the alignment of this class-conditional model $p_\theta(\mathbf{x} \mid y)$ with that of the ground truth data distribution $p_{\text{data}}(\mathbf{x} \mid y)$. We test this hypothesis by studying the effect of density alignment on gradient explanations. To achieve this density alignment, we use an algorithm called score-matching, and propose novel approximations to this algorithm to enable training large-scale models.

Our experiments show that improving the alignment of the implicit density model with the data distribution enhances gradient structure and explanatory power while reducing this alignment has the opposite effect. This also leads us to conjecture that unintended density alignment in standard neural network training may explain the highly structured nature of input-gradients observed in practice. Overall, our finding that input-gradients capture information regarding an implicit generative model implies that we need to re-think their use for interpreting discriminative models.

5.1 Introduction

Input-gradients, or gradients of outputs w.r.t. inputs, are commonly used for the interpretation of deep neural networks (Simonyan et al., 2013). For image classification tasks, an input pixel with a larger input-gradient magnitude is attributed a higher 'importance' value, and the resulting maps are observed to agree with human intuition regarding which input pixels are important for the task at hand (Adebayo et al., 2018). Quantitative studies (Samek et al., 2016; Shrikumar et al., 2017) also show that these importance estimates are meaningful in predicting model response to larger structured perturbations. These results suggest that input-gradients do indeed capture relevant information regarding the underlying model. However in this work, we show that input-gradients can be arbitrarily manipulated using the shift-invariance of softmax without changing the underlying discriminative model, which calls into question the reliability of input-gradient based attribution methods for interpreting arbitrary black-box models.

Given that input-gradients can be arbitrarily structured, the reason for their highly structured and explanatory nature in standard pre-trained models is puzzling. Why are input-gradients relatively well-behaved when they can just as easily be arbitrarily structured, without affecting discriminative model performance? What factors influence input-gradient structure in standard deep neural networks?

To answer these, we consider the connections made between softmax-based discriminative classifiers and generative models (Bridle, 1990; Grathwohl et al., 2020), made by viewing the logits of standard classifiers as un-normalized log-densities. This connection reveals an alternate interpretation of input-gradients, as representing the log-gradients of a class-conditional density model which is implicit within standard softmax-based deep models, which we shall call the *implicit density model*. This connection compels us to consider the following hypothesis: perhaps input-gradients are highly structured because this implicit density model is aligned with the 'ground truth' class-conditional data distribution? The core of this chapter is dedicated to testing the validity of this hypothesis, whether or not input-gradients do become more structured and explanatory if this alignment increases and vice versa.

For the purpose of validating this hypothesis, we require mechanisms to increase or decrease the alignment between the implicit density model and the data distribution. To this end, we consider a generative modelling approach called score-matching, which reduces the density modelling problem to that of local geometric regularization. Hence by using score-matching, we are able to view commonly used geometric regularizers in deep learning as density modelling methods. In practice, the score-matching objective is known for being computationally expensive and unstable to train (Song and Ermon, 2019; Kingma and LeCun, 2010). To this end, we also introduce approximations and regularizers which allow us to use score-matching on practical large-scale discriminative models.

This work is broadly connected to the literature around unreliability of saliency methods. While most such works consider how the explanations for nearly identical images can be arbitrarily different (Dombrowski et al., 2019; Subramanya et al., 2019; Zhang et al., 2020; Ghorbani et al., 2019), our work considers how one may change the model itself to yield arbitrary explanations without affecting discriminative performance. This is similar to Heo et al. (2019) who show this experimentally, whereas we provide an analytical reason for why this happens relating to the shift-invariance of softmax.

The rest of the chapter is organized as follows. We show in § 5.2 that it is trivial to manipulate input-gradients of standard classifiers using the shift-invariance of softmax without affecting the discriminative model. In § 5.3 we state our main hypothesis and describe the details of score-matching, present a tractable approximation for the same that eliminates the need for expensive Hessian computations. § 5.4 revisits other interpretability tools from a density modelling perspective. Finally, § 5.5 presents experimental evidence for the validity of the hypothesis that improved alignment between the implicit density model and the data distribution can improve the structure and explanatory nature of input-gradients.

5.2 Input-Gradients are not Unique

In this section, we show that it is trivial to manipulate input-gradients of discriminative deep networks, using the well-known shift-invariance property of softmax. Here we shall make a distinction between two types of input-gradients: *logit-gradients* and *loss-gradients*. While logit-gradients are gradients of the pre-softmax output of a given class w.r.t. the input, loss-gradients are the gradients of the loss w.r.t. the input. In both cases, we only consider outputs of a single class, usually the target class.

Let $\mathbf{x} \in \mathbb{R}^D$ be a data point, which is the input for a neural network model $f : \mathbb{R}^D \to \mathbb{R}^C$ intended for classification, which produces pre-softmax logits for C classes. The cross-entropy loss function for some class $1 \leq i \leq C,\ i \in \mathbb{N}$ corresponding to an input $\mathbf{x}$ is given by $\ell(f(\mathbf{x}), i) \in \mathbb{R}_+$, which is shortened to $\ell_i(\mathbf{x})$ for convenience. Note that here the loss function subsumes the softmax function as well. The logit-gradients are given by $\nabla_{\mathbf{x}} f_i(\mathbf{x}) \in \mathbb{R}^D$ for class i, while loss-gradients are $\nabla_{\mathbf{x}} \ell_i(\mathbf{x}) \in \mathbb{R}^D$. Let the softmax function be $p(y = i|\mathbf{x}) = \exp(f_i(\mathbf{x}))/\sum_{j=1}^{C} \exp(f_j(\mathbf{x}))$, which we denote as p_i for simplicity. Here, we make the observation that upon adding the same scalar function g to all logits, the logit-gradients can arbitrarily change but the loss values do not.

Observation. *Assume an arbitrary function $g : \mathbb{R}^D \to \mathbb{R}$. Consider another neural network function given by $\tilde{f}_i(\cdot) = f_i(\cdot) + g(\cdot)$, for $0 \leq i \leq C$, for which we obtain $\nabla_{\mathbf{x}} \tilde{f}_i(\cdot) = \nabla_{\mathbf{x}} f_i(\cdot) + \nabla_{\mathbf{x}} g(\cdot)$. For this, the corresponding loss values and loss-gradients are unchanged, i.e.; $\tilde{\ell}_i(\cdot) = \ell_i(\cdot)$ and $\nabla_{\mathbf{x}} \tilde{\ell}_i(\cdot) = \nabla_{\mathbf{x}} \ell_i(\cdot)$ as a consequence of the shift-invariance of softmax.*

This explains how the structure of logit-gradients can be arbitrarily changed: one simply needs to add an arbitrary function g to all logits. This implies that individual logit-gradients $\nabla_{\mathbf{x}} f_i(\mathbf{x})$ and logits $f_i(\mathbf{x})$ are meaningless on their own, and their structure may be uninformative regarding the underlying discriminative model. Despite this, a large fraction of work in interpretable deep learning (Simonyan et al., 2013; Selvaraju et al., 2017; Smilkov et al., 2017; Fong et al., 2019; Srinivas and Fleuret, 2019) uses individual logits and logit-gradients for saliency map computation. We also provide a similar illustration in the supplementary material for the case of loss-gradients, where we show that it is possible for loss-gradients to diverge significantly even when the loss values themselves do not.

These simple observations leave an open question: why are input-gradients highly structured and explanatory when they can just as easily be arbitrarily structured, without affecting discriminative model performance? Further, if input-gradients do not depend strongly on the underlying discriminative function, what aspect of the model do they depend on instead? In the section that follows, we shall consider a generative modelling view of discriminative neural networks that offers insight into the information encoded by logit-gradients.

5.3 Implicit Density Models Within Discriminative Classifiers

Let us consider the following link between generative models and the softmax function. We first define the following joint density on the logits f_i of classifiers: $p_\theta(\mathbf{x}, y = i) = \frac{\exp(f_i(\mathbf{x};\theta))}{Z(\theta)}$, where $Z(\theta)$ is the partition function. We shall henceforth suppress the dependence of f on θ for brevity. Upon using Bayes' rule to obtain $p_\theta(y = i \mid \mathbf{x})$, we observe that we recover the standard softmax function. Thus the logits of discriminative classifiers can alternately be viewed as un-normalized log-densities of the joint distribution. Assuming equiprobable classes, we have $p_\theta(\mathbf{x} \mid y = i) = \frac{\exp(f_i(\mathbf{x}))}{Z(\theta)/C}$, which is the quantity of interest for us. Thus while the logits represent un-normalized log-densities, logit-gradients represent the score function, i.e.; $\nabla_x \log p_\theta(\mathbf{x} \mid y = i) = \nabla_x f_i(\mathbf{x})$, which avoids dependence on the partition function $Z(\theta)$ as it is independent of $\mathbf{x}$.

This viewpoint naturally leads to the following hypothesis, that perhaps the reason for the highly structured and explanatory nature of input-gradients is that the implicit density model $p_\theta(\mathbf{x} \mid y)$ is close to that of the ground truth class-conditional data distribution $p_{\text{data}}(\mathbf{x} \mid y)$? We propose to test this hypothesis explicitly using score-matching as a density modelling tool.

Hypothesis. *(Informal) Improved alignment of the implicit density model to the ground truth class-conditional density model improves input-gradient interpretability via both qualitative and quantitative measures, whereas deteriorating this alignment has the op-*

posite effect.

5.3.1 Score-Matching

Score-matching (Hyvärinen, 2005) is a generative modelling objective that focusses solely on the derivatives of the log density instead of the density itself, and thus does not require access to the partition function $Z(\theta)$. Specifically, for our case we have $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x} \mid y = i) = \nabla_{\mathbf{x}} f_i(\mathbf{x})$, which are the logit-gradients.

Given i.i.d. samples $\mathcal{X} = \{x_i \in \mathbb{R}^D\}$ from a latent data distribution $p_{data}(\mathbf{x})$, the objective of generative modelling is to recover this latent distribution using only samples $\mathcal{X}$. This is often done by training a parameterized distribution $p_\theta(\mathbf{x})$ to align with the latent data distribution $p_{data}(\mathbf{x})$. The score-matching objective instead aligns the gradients of log densities, as given below.

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{p_{data}(\mathbf{x})} \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})\|_2^2 && (5.1) \\
&= \mathbb{E}_{p_{data}(\mathbf{x})} \left(\text{trace}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})) + \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2 \right) + \texttt{const} && (5.2)
\end{aligned}
$$

The above relationship is proved (Hyvärinen, 2005) using integration by parts. This is a consistent objective, *i.e,* $J(\theta) = 0 \iff p_{data} = p_\theta$. This approch is appealing also because this reduces the problem of generative modelling to that of regularization of the local geometry of functions, i.e.; the resulting terms only depend on the point-wise gradients and Hessian-trace.

5.3.2 Efficient estimation of Hessian-trace

In general, equation 5.2 is intractable for high-dimensional data due to the Hessian trace term. To address this, we can use the Hutchinson's trace estimator (Hutchinson, 1990) to efficiently compute an estimate of the trace by using random projections, which is given by: $\text{trace}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})) = \mathbb{E}_{\mathbf{v} \sim \mathcal{N}(0, \mathtt{I})}\ \mathbf{v}^\intercal\ \nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})\ \mathbf{v}$. This estimator has been previously applied to score-matching (Song et al., 2019), and can be computed efficiently using Pearlmutter's trick (Pearlmutter, 1994). However, this trick still requires **two backward passes** for a single monte-carlo sample, which is computationally expensive. To further improve computational efficiency, we introduce the following approximation to Hutchinson's estimator using a Taylor series expansion, which applies to small values of $\sigma \in \mathbb{R}$.

$$\begin{aligned}\mathbb{E}_{\mathbf{v}\sim\mathcal{N}(0,\mathtt{I})}\ \mathbf{v}^{\mathtt{T}}\nabla^2_{\mathbf{x}}\log p_\theta(\mathbf{x})\mathbf{v} &\approx \frac{2}{\sigma^2}\,\mathbb{E}_{\mathbf{v}\sim\mathcal{N}(0,\sigma^2\mathtt{I})}\left(\log p_\theta(\mathbf{x}+\mathbf{v}) - \log p_\theta(\mathbf{x}) - \nabla_x \log p_\theta(\mathbf{x})^{\mathtt{T}}\mathbf{v}\right) \\ &= \frac{2}{\sigma^2}\,\mathbb{E}_{\mathbf{v}\sim\mathcal{N}(0,\sigma^2\mathtt{I})}\left(\log p_\theta(\mathbf{x}+\mathbf{v}) - \log p_\theta(\mathbf{x})\right) \qquad (5.3)\end{aligned}$$

Note that equation 5.3 involves a difference of log probabilities, which is independent of the partition function. For our case, $\log p_\theta(\mathbf{x}+\mathbf{v}|y=i) - \log p_\theta(\mathbf{x}|y=i) = f_i(\mathbf{x}+\mathbf{v}) - f_i(\mathbf{x})$. We have thus considerably simplified and speeded-up the computation of the Hessian trace term, which now can be approximated with **no backward passes**, but using only a single additional forward pass. We present details regarding the variance of this estimator in the supplementary material. A concurrent approach (Pang et al., 2020) also presents a similar algorithm, however it is applied primarily to Noise Contrastive Score Networks (Song and Ermon, 2019) and Denoising Score Matching (Vincent, 2011), whereas we apply it to vanilla score-matching on discriminative models.

5.3.3 Stabilized Score-matching

In practice, a naive application of score-matching objective is unstable, causing the Hessian-trace to collapse to negative infinity. This occurs because the finite-sample variant of equation 5.1 causes the model to 'overfit' to a mixture-of-diracs density, which places a dirac-delta distribution at every data point. Gradients of such a distribution are undefined, causing training to collapse. To overcome this, regularized score-matching (Kingma and LeCun, 2010) and noise conditional score networks (Song and Ermon, 2019) propose to add noise to inputs for score-matching to make the problem well-defined. However, this did not help for our case. Instead, we use a heuristic where we add a small penalty term proportional to the square of the Hessian-trace. This discourages the Hessian-trace becoming too large, and thus stabilizes training.

5.4 Implications of the Density Modelling Viewpoint

In the previous section we related input-gradients to the implicit density model, thus linking gradient interpretability to density modelling through our hypothesis. In this section, we consider two other interpretability tools: activity maximization and the pixel perturbation test, and show how these can interpreted from a density modelling perspective. These perspectives also enable us to draw parallels between score-matching and adversarial training.

5.4.1 Activity Maximization as Sampling from the Implicit Density Model

The canonical method to obtain samples from score-based generative models is via Langevin sampling (Welling and Teh, 2011; Song and Ermon, 2019), which involves performing gradient ascent on the density model with noise added to the gradients. Without this added noise, the algorithm recovers the modes of the density model.

We observe that activity maximization algorithms used for neural network visualizations are remarkably similar to this scheme. For instance, Simonyan et al. (2013) recover inputs which maximize the logits of neural networks, thus exactly recovering the modes of the implicit density model. Similarly, deep-dream-like methods (Mahendran and Vedaldi, 2016; Nguyen et al., 2016; **?**) extend this by using "image priors" to ensure that the resulting samples are closer to the distribution of natural images, and by adding structured noise to the gradients in the form of jitter, to obtain more visually pleasing samples. From the density modelling perspective, we can alternately view these visualization techniques as biased sampling methods for score-based density models trained on natural images. However, given the fact that they draw samples from the implicit density model, their utility in interpreting discriminative models may be limited.

5.4.2 Pixel Perturbation as a Density Ratio Test

A popular test for saliency map evaluation is based on pixel perturbation (Samek et al., 2016). This involves first selecting the least-relevant (or most-relevant) pixels according to a saliency map representation, 'deleting' those pixels and measuring the resulting change in output value. Here, deleting a pixel usually involves replacing the pixel with a non-informative value such as a random or a fixed constant value. A good saliency method identifies those pixels as less relevant whose deletion does not cause a large change in output value.

We observe that this change in outputs criterion is identical to the density ratio, *i.e.*, $\log\left(p_\theta(\mathbf{x}+\mathbf{v}|y=i)/p_\theta(\mathbf{x}|y=i)\right) = f_i(\mathbf{x}+\mathbf{v}) - f_i(\mathbf{x})$. Thus when logits are used for evaluating the change in outputs (Samek et al., 2016; Ancona et al., 2018), the pixel perturbation test exactly measures the density ratio between the perturbed image and the original image. Thus if a perturbed image has a similar density to that of the original image under the implicit density model, then the saliency method that generated these perturbations is considered to be explanatory. Similarly, Fong et al. (2019) optimize over this criterion to identify pixels whose removal causes minimal change in logit activity, thus obtaining perturbed images with a high implicit density value similar to that of activity maximization. Overall, this test captures sensitivity of the implicit density model, and not the underlying discriminative model which we wish to interpret. We thus recommend that the pixel perturbation test always be used in conjunction with

either the change in output probabilities, or the change in the accuracy of classification, rather than the change in logits.

5.4.3 Connecting Score-Matching to Adversarial Training

Recent works in adversarial machine learning (Etmann et al., 2019; Engstrom et al., 2019; Santurkar et al., 2019; Kaur et al., 2019; Ross and Doshi-Velez, 2017) have observed that saliency map structure and samples from activation maximization are more perceptually aligned for adversarially trained models than for standard models. However it is unclear from these works why this occurs. Separate from this line of work, Chalasani et al. (2018) also connect regularization of a variant of integrated gradients with adversarial training, suggesting a close interplay between the two.

We notice that these properties are shared with score-matched models, or models trained such that the implicit density model is aligned with the ground truth. Further, we note that both score-matching and adversarial training are often based on local geometric regularization, usually involving regularization of the gradient-norm (Ross and Doshi-Velez, 2017; Jakubovitz and Giryes, 2018), and training both the discriminative model and the implicit density model (Grathwohl et al., 2020) has been shown to improve adversarial robustness. From these results, we can conjecture that training the implicit density model via score-matching may have similar outcomes as adversarial training. We leave the verification and proof of this conjecture to future work.

5.5 Experiments

In this section, we present experimental results to show the efficacy of score-matching and the validation of the hypothesis that density alignment influences the gradient explanation quality. For experiments, we shall consider the CIFAR100 dataset. We present experiments with CIFAR10 in the supplementary section. Unless stated otherwise, the network structure we use shall be a 18-layer ResNet that achieves 78.01% accuracy on CIFAR100, and the optimizer used shall be SGD with momentum. All models use the softplus non-linearity with $\beta = 10$, which is necessary to ensure that the Hessian is non-zero for score-matching. Before proceeding with our experiments, we shall briefly introduce the score-matching variants we shall be using for comparisons.

Score-Matching We propose to use the score-matching objective as a regularizer in neural network training to **increase** the alignment of the implicit density model to the ground truth, as shown in equation 5.4, with the stability regularizer discussed in §5.3.3. For this, we use a regularization constant $\lambda = 1e - 3$. This model achieves 72.20% accuracy on the test set, which is a drop of about 5.8% compared to the original model. In the supplementary material, we perform a thorough hyper-parameter sweep

and show that it is possible to obtain better performing models.

$$h(\mathbf{x}) := \frac{2}{\sigma^2}\mathbb{E}_{\mathbf{v}\sim\mathcal{N}(0,\sigma^2\mathrm{I})}\left(f_i(\mathbf{x}+\mathbf{v}) - f_i(\mathbf{x})\right)$$

$$\underbrace{\ell_{reg}(f(\mathbf{x}),i)}_{\text{regularized loss}} = \underbrace{\ell(f(\mathbf{x}),i)}_{\text{cross-entropy}} + \lambda\left(\underbrace{\overbrace{h(\mathbf{x})}^{\text{Hessian-trace}} + \frac{1}{2}\overbrace{\|\nabla_{\mathbf{x}} f_i(\mathbf{x})\|_2^2}^{\text{gradient-norm}}}_{\text{score-matching}} + \underbrace{\overbrace{\mu}^{10^{-4}} h^2(\mathbf{x})}_{\text{stability regularizer}}\right) \tag{5.4}$$

Anti-score-matching We would like to have a tool that can **decrease** the alignment between the implicit density model and the ground truth. To enable this, we propose to maximize the hessian-trace, in an objective we call *anti-score-matching*. For this, we shall use a the clamping function on hessian-trace, which ensures that its maximization stops after a threshold is reached. We use a threshold of $\tau = 1000$, and regularization constant $\lambda = 1e-4$. This model achieves an accuracy of 74.87%.

Gradient-Norm regularization We propose to use gradient-norm regularized models as another baseline for comparison, using a regularization constant of $\lambda = 1e-3$. This model achieves an accuracy of 76.60%.

5.5.1 Evaluating the Efficacy of Score-Matching and Anti-Score-Matching

Here we demonstrate that training with score-matching / anti-score-matching is possible, and that such training improves / deteriorates the quality of the implicit density models respectively as expected.

Density Ratios

One way to characterize the generative behaviour of models is to compute likelihoods on data points. However this is intractable for high-dimensional problems, especially for un-normalized models. We observe although that the densities $p(\mathbf{x} \mid y = i)$ themselves are intractable, we can easily compute density ratios $p(\mathbf{x}+\eta \mid y = i)/p(\mathbf{x} \mid y = i) = \exp(f_i(\mathbf{x}+\eta) - f_i(\mathbf{x}))$ for a random noise variable η. Thus, we propose to plot the graph of density ratios locally along random directions. These can be thought of as local cross-sections of the density sliced at random directions. We plot these values for gaussian noise η for different standard deviations, which are averaged across points in the entire dataset.

In Figure 5.1, we plot the density ratios upon training on the CIFAR100 dataset. We observe that the baseline model assigns *higher* density values to noisy inputs than

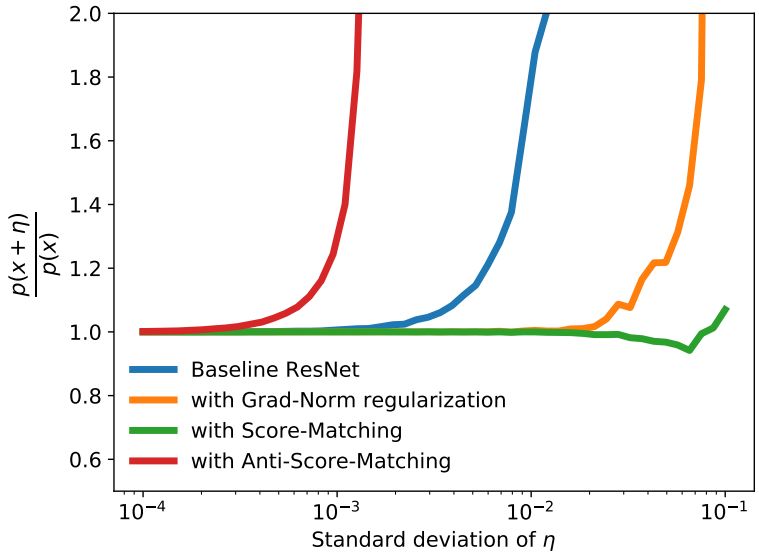


Figure 5.1 – Plots of density ratios representing local density profiles across varying levels of noise added to the input (lower is better). We observe that score-matched model is robust to a larger range of noise values, while anti-score-matching is very sensitive even to small amounts of noise.

real inputs. With anti-score-matching, we observe that the density profile grows still steeper, assigning higher densities to inputs with smaller noise. Gradient-norm regularized models and score-matched models improve on this behaviour, and are robust to larger amounts of noise added. Thus we are able to obtain penalty terms that can both improve and deteriorate the density modelling behaviour within discriminative models.

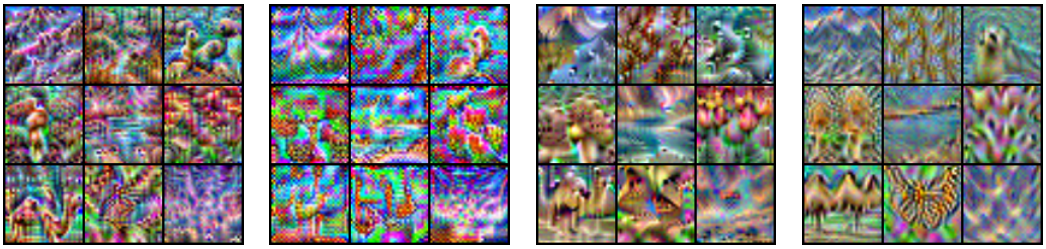
Sample Quality

We are interested in recovering modes of our density models while having access to only the gradients of the log density. For this purpose, we apply gradient ascent on the log probability $\log p(\mathbf{x} \mid y = i) = f_i(\mathbf{x})$, similar to activity maximization. Our results are shown in Figure 5.2. We observe that samples from the score-matched and gradient-norm regularized models are significantly less noisy than other models.

We also propose to qualitatively measure the sample quality using the GAN-test approach (Shmelkov et al., 2018). This test proposes to measure the discriminative accuracy of generated samples via an independently trained discriminative model. In contrast with more popular metrics such as the inception-score, this captures sample quality rather than diversity, which is what we are interested in. We show the results in table 5.1, which confirms the qualitative trend seen in samples above. Surprisingly, we find that gradient-norm regularized models perform better than score-matched models.

Model	GAN-test (%)
Baseline ResNet	59.47
+ Anti-Score-Matching	16.40
+ Gradient Norm-regularization	**80.07**
+ Score-Matching	72.75

Table 5.1 – GAN-test scores (higher is better) of class-conditional samples generated from various ResNet-18 models (see § 5.5.1). We observe that samples from gradient-norm regularized models and score-matched models achieve much better accuracies than the baselines and anti-score-matched models.



(a) Baseline ResNet (b) With anti score-matching (c) With Gradient-norm regularization (d) With score-matching

Figure 5.2 – Samples generated from various models by performing gradient ascent on random inputs (see § -5.5.1). While none of the generated samples are realistic, samples obtained from score-matched and gradient-norm regularized models are smoother and less noisy.

This implies that such models are able to implicitly perform density modelling without being explicitly trained to do so. We leave further investigation of this phenomenon to future work.

5.5.2 Evaluating the Effect of Density Alignment on Gradient Explanations

Here we shall evaluate the gradient explanations of various models. First, we shall look at quantitative results on a discriminative variant of the pixel perturbation test. Second, we visualize the gradient maps to assess qualitative differences between them.

Quantitative Results on Discriminative Pixel Perturbation

As noted in 5.4.2, it is recommended to use the pixel perturbation test using accuracy changes, and we call this variant as *discriminative pixel perturbation*. We select the least relevant pixels and replace them with the mean pixel value of the image, note down the

accuracy of the model on the resulting samples. We note that this test is only used so far to compare different saliency methods for the same underlying model. However, we here seek to compare saliency methods across models. For this we consider two experiments. First, we perform the pixel perturbation experiment with each of the four trained models on their own input-gradients and plot the results in Figure 5.3a. These results indicate that the input-gradients of score-matched and gradient-norm regularized models are better equipped to identify least relevant pixels in this model. However, it is difficult to completely disentangle the robustness benefits of such score-matched models against improved identification of less relevant pixels through such a plot.

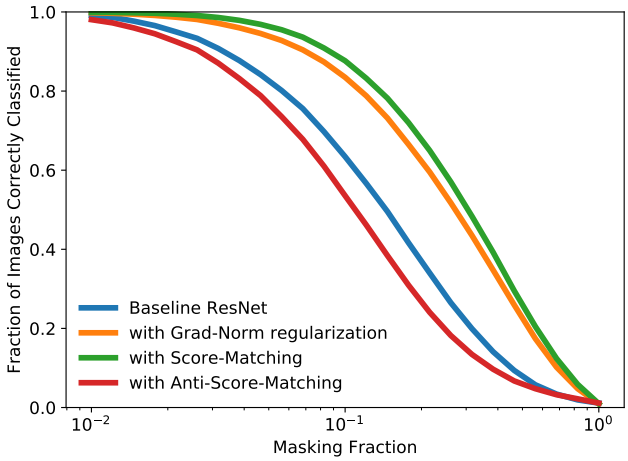
To this end, we conduct a second experiment in Figure 5.3b, where we use input-gradients obtained from these four trained models to explain the same standard baseline ResNet model. This disentangles the robustness of different models as inputs to the same model is perturbed in all cases. Here also we find that gradients from score-matched and gradient-norm regularized models explain behavior of standard baseline models better than the gradients of the baseline model itself. Together, these tests show that training with score-matching indeed produces input-gradients that quantitatively more explanatory than baseline models.

Qualitative Gradient Visualizations

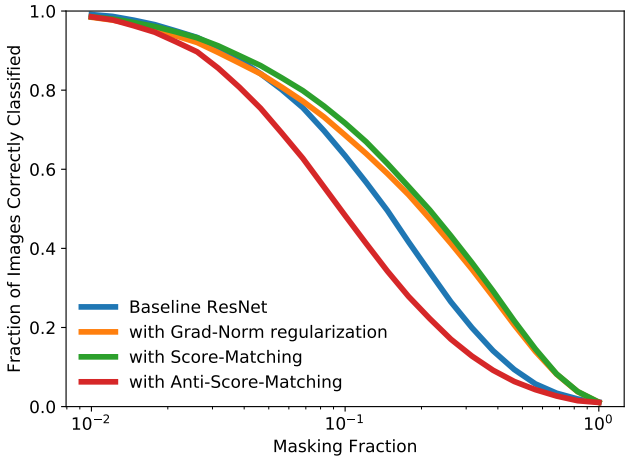
We visualize the structure of logit-gradients of different models in Figure 5.4. We observe that gradient-norm regularized model and score-matched model have highly perceptually aligned gradients, when compared to the baseline and anti-score-matched gradients, corroborating the quantitative results.

5.6 Conclusion

In this chapter, we investigated the cause for the highly structured and explanatory nature of input-gradients in standard pre-trained models, and showed that alignment of the implicit density model with the ground truth data density is a possible cause. This density modelling interpretation enabled us to view canonical approaches in interpretability such as gradient-based saliency methods, activity maximization and the pixel perturbation test through a density modelling perspective, showing that these capture information relating to the implicit density model, not the underlying discriminative model which we wish to interpret. This calls for a need to re-think the role of these tools in the interpretation of discriminative models. For practitioners, we believe it is best to avoid usage of logit gradient-based tools, for interpretability. If unavoidable, it is recommended to use only gradient-norm regularized or score-matched models, as input-gradients of these models produce more reliable estimates of the gradient of the underlying distribution. As our experiments show, these may be a useful tool even



(a) Models evaluated with their own gradients

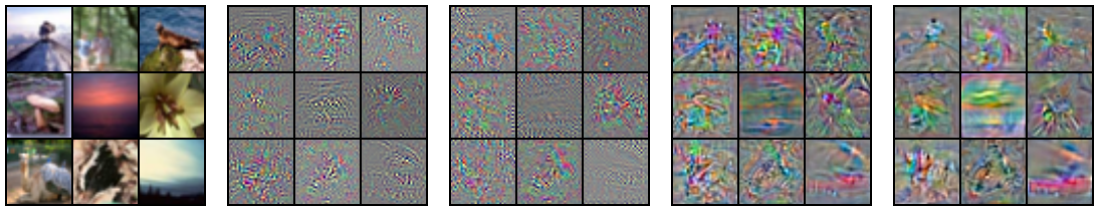


(b) Baseline ResNet evaluated with gradients of *different* models

Figure 5.3 – Discriminative pixel perturbation results (higher is better) on the CIFAR100 dataset (see § 5.5.2). We see that score-matched and gradient-norm regularized models best explain model behaviour in both cases, while the anti-score-matched model performs the worst. This agrees with the hypothesis (stated in § 5.3) that alignment of implicit density models improves gradient explanations and vice versa.

though they are not directly related to the discriminative model.

However, our work still does not answer the question of why pre-trained models may have their implicit density models aligned with ground truth in the first place. One possible reason could be the the presence of an implicit gradient norm regularizer in standard SGD, similar to that shown independently by Barrett and Dherin (2020). Another open question is to understand why gradient-norm regularized models are able to perform implicit density modelling as observed in our experiments in § 5.5.1, which



(a) Input Image (b) Baseline ResNet (c) With Anti score-matching (d) With Gradient-norm regularization (e) With Score-matching

Figure 5.4 – Visualization of input-gradients of different models. We observe that gradients of score-matched and gradient-norm regularized models are more perceptually aligned than the others, with the gradients of the anti-score-matched model being the noisiest. This qualitatively verifies the hypothesis stated in § 5.3.

lead to improved gradient explanations.

Appendix

5.7 Fooling Gradients is simple

Observation. *Assume an arbitrary function $g : \mathbb{R}^D \to \mathbb{R}$. Consider another neural network function given by $\tilde{f}_i(\cdot) = f_i(\cdot) + g(\cdot)$, for $0 \leq i \leq C$, for which we obtain $\nabla_{\mathbf{x}} \tilde{f}_i(\cdot) = \nabla_{\mathbf{x}} f_i(\cdot) + \nabla_{\mathbf{x}} g(\cdot)$. For this, the corresponding loss values and loss-gradients are unchanged, i.e.; $\tilde{\ell}_i(\cdot) = \ell_i(\cdot)$ and $\nabla_{\mathbf{x}} \tilde{\ell}_i(\cdot) = \nabla_{\mathbf{x}} \ell_i(\cdot)$.*

Proof. The following expressions relate the loss and neural network function outputs, for the case of cross-entropy loss and usage of the softmax function.

$$\ell_i(\mathbf{x}) = -f_i(\mathbf{x}) + \log\left(\sum_{j=1}^{C} \exp(f_j(\mathbf{x}))\right) \tag{5.5}$$

$$\nabla_{\mathbf{x}} \ell_i(\mathbf{x}) = -\nabla_{\mathbf{x}} f_i(\mathbf{x}) + \sum_{j=1}^{C} p_j \nabla_{\mathbf{x}} f_j(\mathbf{x}) \tag{5.6}$$

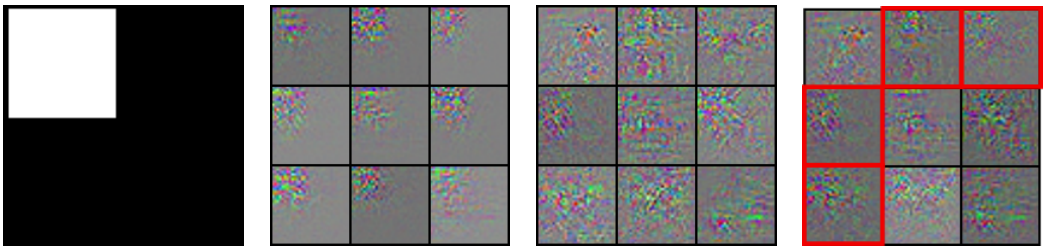
Upon replacing f_i with $\tilde{f}_i = f_i + g$, the proof follows. □

5.7.1 Manipulating Loss-Gradients

Here, we show how we can also change loss-gradients arbitrarily without significantly changing the loss values themselves. In this case, the trick is to add a high frequency low amplitude sine function to the loss.

Observation. *Consider $g(\mathbf{x}) = \epsilon \sin(m\mathbf{x})$, and $\tilde{\ell}_i(\mathbf{x}) = \ell_i(\mathbf{x}) + g(\mathbf{x})$, for $\epsilon, m \geq \mathbb{R}_+$ and $\mathbf{x} \in \mathbb{R}^D$. Then, it is easy to see that $|\tilde{\ell}_i(\mathbf{x}) - \ell_i(\mathbf{x})| \leq \epsilon$, and $\|\nabla_{\mathbf{x}} \tilde{\ell}_i(\mathbf{x}) - \nabla_{\mathbf{x}} \ell_i(\mathbf{x})\|_1 \leq m \times \epsilon \times D$.*

Thus two models with losses differing by some small ϵ can have gradients differing by $m \times \epsilon \times D$. For $m \to \infty$ and a fixed ϵ, the gradients can diverge significantly. Thus,



(a) Fooling Mask (b) Fooled logit-gradients (c) Loss-gradients (d) Temperature scaled loss-gradients

Figure 5.5 – Results of fooling neural network logit-gradients. Given a mask **(a)**, we are able to fool logit-gradients **(b)**. We observe that loss-gradients **(c)** are not affected, however they can change to adhere to the mask upon using a high-temperature softmax **(d)**, as indicated by the areas in red.

loss-gradients are also unreliable, as two models with very similar loss landscapes and hence discriminative abilities, can have drastically different loss-gradients.

This simple illustration highlights the fact that gradients of high-dimensional black-box models are not well-behaved in general, and this depends on both the model smoothness and the high-dimensionality of the inputs. Further, loss values and loss-gradients for highly confident samples are close to zero. Thus any external noise added (due to stochastic training, for instance) can easily dominate the loss-gradient terms even when smoothness conditions (small m) are enforced.

5.7.2 Experiments on Fooling Gradient Explanations

Here we present experimental evidence to support the claim that fooling input-gradients is simple. First, we show how loss-gradients are unchanged when logit-gradients are fooled. Second, we show that how loss-gradients can also be fooled by simply increasing the temperature parameter within softmax. Our experiments are performed on the CIFAR100 dataset, using a 11-layer VGG network.

Given a normalized unsigned saliency map $\mathbf{s} = |\nabla_{\mathbf{x}} f_i|/(1^{\mathrm{T}}|\nabla_{\mathbf{x}} f_i|)$ and a desired normalized binary mask structure $\mathbf{m}$, the saliency fooling algorithm (Heo et al., 2019) consists of the following objective function.

$$\mathcal{L}_{fool}(\mathbf{s}, \mathbf{m}) = \mathbb{E}_{\mathbf{x}} \|\mathbf{s} - \mathbf{m}\|^2 \tag{5.7}$$

We add this as a regularizer along with the standard cross-entropy loss and fine-tune a pre-trained VGG classifier. We assume the mask structure given in Figure 5.5a, which comprises of a 15×15 white region. Assuming a uniform distribution of logit-gradients over pixels, one would expect 22% of the total energy of unsigned gradients to occur

in the top left region. Upon optimizing the fooling objective, we observe that we are indeed able to fool logit-gradients, with these having 83.85% of the total energy in only the top left areas, as shown in Figure 5.5b. However, we note that loss-gradients are not fooled, with an average energy of only 48.14% in the unmasked areas. Our attempts at fooling loss-gradients in a similar manner were unsuccessful: either the training collapsed completely or fooling failed to occur.

Our second experiment involves testing whether the loss-gradients can be fooled for low probability classes. To test this, use a high temperature constant ($T = 1e3$) with softmax, for the fooled model above. Upon doing so, we see that the loss-gradients are also altered, with an average energy of 60.17% in the top left region, up from 48.14% as shown in Figure 5.5d. This provides experimental validation for our theory.

5.7.3 Implications for Saliency Regularization Methods

We mention a few important points of discussion here. First, the example shown above is simpler than the theory permits, i.e.; in general the masks can vary with the input, whereas here we use the same mask for all inputs for simplicity. Second, we note that similar schemes have been presented (Ross et al., 2017; Erion et al., 2019) to align loss-gradients with hand-crafted masks in order to inject domain knowledge into models. While these schemes may achieve their goals in certain cases, our analysis shows that these cannot be applied universally, as loss-gradients may change by changing the underlying loss values by only a small amount.

5.8 Score-Matching Approximation

We consider the approximation derived for the estimator of the Hessian trace, which is first derived from Hutchinson's trace estimator Hutchinson (1990). We replace $\log p_\theta(\mathbf{x})$ terms used in the main text with $f(\mathbf{x})$ terms here for clarity. The Taylor series trick for approximating the Hessian-trace is given below.

$$\begin{aligned}
\mathbb{E}_{\mathbf{v}\sim\mathcal{N}(0,\mathtt{I})}\ \mathbf{v}^{\mathtt{T}}\nabla^2_{\mathbf{x}} f(\mathbf{x})\mathbf{v} &= \frac{1}{\sigma^2}\ \mathbb{E}_{\mathbf{v}\sim\mathcal{N}(0,\sigma^2\mathtt{I})}\mathbf{v}^{\mathtt{T}}\nabla^2_{\mathbf{x}} f(\mathbf{x})\mathbf{v} \\
&= \frac{2}{\sigma^2}\ \mathbb{E}_{\mathbf{v}\sim\mathcal{N}(0,\sigma^2\mathtt{I})}\Big(f(\mathbf{x}+\mathbf{v}) - f(\mathbf{x}) - \nabla_x f(\mathbf{x})^{\mathtt{T}}\mathbf{v} + \mathcal{O}(\sigma^3)\Big)
\end{aligned} \tag{5.8}$$

As expected, the approximation error vanishes in the limit of small σ. Let us now consider the finite sample variants of this estimator, with N samples. We shall call this

the *Taylor Trace Estimator*.

$$\text{Taylor Trace Estimator (TTE)} = \frac{2}{N\sigma^2}\sum_{i=1}^{N}\Big(f(\mathbf{x}+\mathbf{v}_i)-f(\mathbf{x})\Big) \quad \text{s.t.} \quad \mathbf{v}_i \sim \mathcal{N}(0,\sigma^2\mathtt{I}) \tag{5.9}$$

We shall henceforth suppress the dependence on i for brevity. For this estimator, we can compute its variance for quadratic functions f, where higher-order Taylor expansion terms are zero. We make the following observation.

Observation. *For quadratic functions f, the variance of the Taylor Trace Estimator is greater than the variance of the Hutchinson estimator by an amount at most equal to $4\sigma^{-2}\|\nabla_{\mathbf{x}} f(\mathbf{x})\|^2$.*

Proof.

$$\begin{aligned}
\text{Var(T.T.E.)} &= \frac{1}{\sigma^4}\,\mathbb{E}_v\left(\frac{2}{N}\sum_{i=1}^{N}\Big(f(\mathbf{x}+\mathbf{v})-f(\mathbf{x})\Big)-\mathbb{E}_v\mathbf{v}^{\mathrm{T}}\nabla^2_{\mathbf{x}}f(\mathbf{x})\mathbf{v}\right)^2 \\
&= \frac{1}{\sigma^4}\,\mathbb{E}_v\Bigg(\frac{2}{N}\sum_{i=1}^{N}\Big(f(\mathbf{x}+\mathbf{v})-f(\mathbf{x})\Big)-\frac{1}{N}\sum_{i=1}^{N}\mathbf{v}^{\mathrm{T}}\nabla^2_{\mathbf{x}}f(\mathbf{x})\mathbf{v} \\
&\quad +\frac{1}{N}\sum_{i=1}^{N}\mathbf{v}^{\mathrm{T}}\nabla^2_{\mathbf{x}}f(\mathbf{x})\mathbf{v}-\mathbb{E}_v\mathbf{v}^{\mathrm{T}}\nabla^2_{\mathbf{x}}f(\mathbf{x})\mathbf{v}\Bigg)^2 \\
&= \frac{1}{\sigma^4}\,\mathbb{E}_v\left(\frac{2}{N}\sum_{i=1}^{N}\Big(f(\mathbf{x}+\mathbf{v})-f(\mathbf{x})\Big)-\frac{1}{N}\sum_{i=1}^{N}\mathbf{v}^{\mathrm{T}}\nabla^2_{\mathbf{x}}f(\mathbf{x})\mathbf{v}\right)^2 \\
&\quad +\frac{1}{\sigma^4}\mathbb{E}_v\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{v}^{\mathrm{T}}\nabla^2_{\mathbf{x}}f(\mathbf{x})\mathbf{v}-\mathbb{E}_v\mathbf{v}^{\mathrm{T}}\nabla^2_{\mathbf{x}}f(\mathbf{x})\mathbf{v}\right)^2
\end{aligned}$$

Thus we have decomposed the variance of the overall estimator into two terms: the first captures the variance of the Taylor approximation, and the second captures the variance of the Hutchinson estimator.

Considering only the first term, i.e.; the variance of the Taylor approximation, we have:

$$\begin{aligned}
\frac{1}{N\sigma^4}\,\mathbb{E}_v\left(2\sum_{i=1}^{N}\Big(f(\mathbf{x}+\mathbf{v})-f(\mathbf{x})\Big)-\sum_{i=1}^{N}\mathbf{v}^{\mathrm{T}}\nabla^2_{\mathbf{x}}f(\mathbf{x})\mathbf{v}\right)^2 &= \frac{4}{N\sigma^4}\,\mathbb{E}_v\left(\sum_{i=1}^{N}\nabla_{\mathbf{x}}f(\mathbf{x})^T\mathbf{v}\right)^2 \\
&\leq \frac{4}{\sigma^4}\|\nabla_{\mathbf{x}}f(\mathbf{x})\|^2\mathbb{E}_v\|\mathbf{v}\|^2 \\
&= 4\sigma^{-2}\|\nabla_{\mathbf{x}}f(\mathbf{x})\|^2
\end{aligned}$$

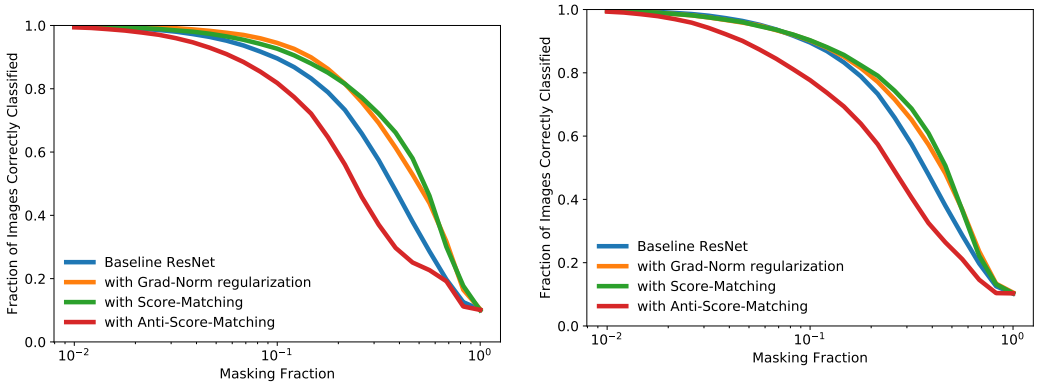
The intermediate steps involve expanding the summation, noticing that pairwise terms cancel, and applying the Cauchy-Schwartz inequality. □

Thus we have a trade-off: a large σ results in lower estimator variance but a large Taylor approximation error, whereas the opposite is true for small σ. However for functions with small gradient norm, both the estimator variance and Taylor approximation error is small for small σ. We note that when applied to score-matching Hyvärinen (2005), the gradient norm of the function is also minimized. This implies that in practice, the gradient norm of the function is likely to be low, thus resulting in a small estimator variance even for small σ. The variance of the Hutchinson estimator is given below for reference Hutchinson (1990); Avron and Toledo (2011):

$$\text{Var(Hutchinson)} = \frac{2}{N}\|\nabla^2_{\mathbf{x}} f(\mathbf{x})\|^2_F$$

5.9 Evaluating Effect of Score-Matching on Gradient Explanations on CIFAR10

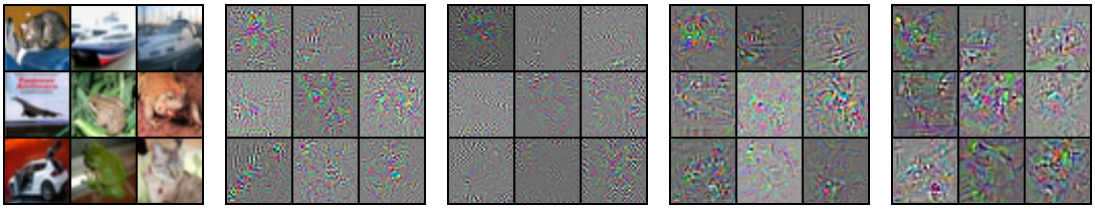
We repeat the pixel perturbation experiments on the CIFAR10 dataset and we observe similar qualitative trends. In both cases, we observe that score-matched and gradient norm regularized models have more explanatory gradients, while anti-score-matched model contains the least explanatory gradients. We also present visualization results of input-gradients of various models for reference.



(a) Models evaluated with their own gradients

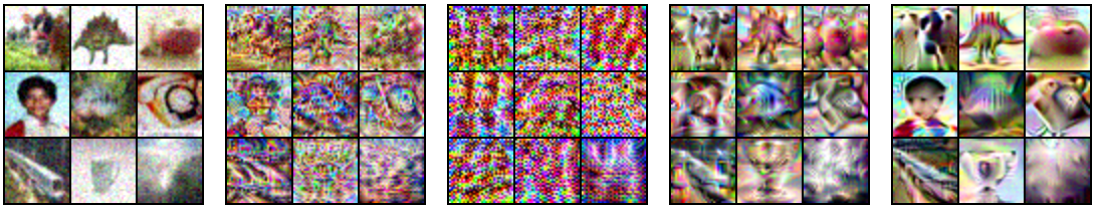
(b) Baseline ResNet evaluated with gradients of *different* models

Figure 5.6 – Discriminative pixel perturbation results (higher is better) on the CIFAR10 dataset (see § 5.5.2). We see that score-matched and gradient-norm regularized models best explain model behaviour in both cases, while the anti-score-matched model performs the worst. This agrees with the hypothesis (stated in § 5.3) that alignment of implicit density models improves gradient explanations and vice versa.



(a) Input Image (b) Baseline ResNet (c) With Anti score-matching (d) With Gradient-norm regularization (e) With Score-matching

Figure 5.7 – Visualization of input-gradients of different models. We observe that gradients of score-matched and gradient-norm regularized models are more perceptually aligned than the others, with the gradients of the anti-score-matched model being the noisiest. This qualitatively verifies the hypothesis stated in § 5.3.



(a) Noisy Image (b) Baseline ResNet (c) With Anti score-matching ResNet (d) With Gradient-norm regularization (e) With score-matching

Figure 5.8 – 'Denoised' samples generated from models by performing gradient ascent on inputs perturbed with noise ($\sigma = 0.1$). Sample quality drastically improves with score-matching.

5.10 Denoising via Implicit Density Models on CIFAR100

To test the quality of implicit density models, we also run a 'denoising' experiment, where we perform Langevin sampling with data points perturbed with small noise, instead of random points as is done usually. The modes of an ideal generative model lie near clean, un-noised data, thus motivating this experiment. Figure 5.8 shows that denoised samples of score-matched models are significantly more realistic than the rest.

5.11 Hyper-parameter Sweep on Score-Matched Training

We present results on a hyper-parameter sweep on the λ and μ parameters of score-matching, where we provide both test-set accuracy on CIFAR100 and the corresponding GAN-test scores. We find upon performing a hyper-parameter sweep that $\lambda = 1e-5$ and $\mu = 1e-3$ seems to perform the best, whereas in the main chapter we present results for $\lambda = 1e-3$ and $\mu = 1e-4$. It is possible that changing the training schedule by increasing the number of epochs or learning rate may further improve these results,

but we did not explore that here.

$\lambda\downarrow/\mu\rightarrow$	$1e-2$	$1e-3$	$1e-4$	$1e-5$
$1e-2$	48.68%/51.60%	64.57%/58.90%	64.75%/76.46%	9.08%/0.97%
$1e-3$	64.64%/56.78%	71.37%/40.72%	72.34%/73.39%	34.46%/3.3%
$1e-4$	69.85%/41.30%	73.97%/72.07%	75.65%/79.39%	72.97%/61.52%
$1e-5$	73.29%/68.94%	**75.37**%/**85.64**%	76.40%/63.96%	74.80%/78.41%
$1e-6$	75.43%/82.81%	75.90%/66.11%	76.77%/65.91%	75.91%/65.52%

Table 5.2 – Results of a hyper-parameter sweep on λ and μ. The numbers presented are in the format (accuracy % / GAN-test %)

6 Conclusions, Limitations & Open Problems

In this thesis, we have studied the functional behaviour of neural network models using gradient-based methods for two related tasks. The first task was knowledge transfer, which involved teaching a student model to mimic the functional behaviour of a teacher model, where we used input-gradients for efficient transfer. The second task was post-hoc interpretation, which involved capturing the functional behaviour in pre-trained neural network models and communicating these to humans, for which we used the full-gradient representation, which captures complete information regarding the computation for an individual sample. We also found this to be useful for knowledge transfer as well as regularization. Finally, we took a step back asked why gradient based interpretation methods yield highly structured maps in the first place, and we found that implicit generative modelling capability may be a contributing factor.

One limitation of using gradient-based losses in general is that they are difficult to optimize, possibly due to complex nature of the loss landscape. We discussed this issue both in Chapters 2 and 4, where we found that using larger student models resulted in easier optimization in terms of being able to achieve a lower loss value at the end of optimization. One direction for future work in this area is to investigate the fundamental root of this issue, and to propose optimization methods that are better able to better optimize such losses, or equivalent losses that work well with conventional optimization methods.

An important drawback of the current state of post-hoc interpretability literature particularly with saliency maps is that methods are often evaluated only for faithfulness, i.e., whether they capture the underlying function behaviour well. However, the main goal of interpretability is to help human domain experts understand the machine learning models they intend to work with, and evaluations which measure this are typically not performed in saliency map literature (Doshi-Velez and Kim, 2017). Future work must thus identify suitable protocols to measure the efficacy of interpretability methods with human test subjects, in a way that is decoupled with model behaviour itself. We sus-

pect that work in this area must interface with fields such as psychology and education research in order to make formal statements about the level of model understanding of humans and judge whether this has improved as a result of using interpretability methods.

Another limitation of gradient-based interpretability methods is that they capture pixel-wise importances in images, and thus are always approximate representations of model behaviour, as typical models do not process images pixel-wise (like Generalized Additive Models), but process the entire image at once. One direction for future research is to investigate for which model classes are gradient-based interpretability methods faithful representations of model behaviour. If such model classes are successfully identified, then the reliability of gradient-based explanations can be estimated by computing a "distance" from this interpretable model class to the particular model we have at hand.

The core of any study on implicit density models relies on an efficient method to train such models. In this thesis, we proposed simplified score-matching strategy using a finite-difference approach. The drawbacks of this strategy are the usage of noise sampling for finite differences, and for high dimensional inputs, one needs to draw a large number of samples for accurate Hessian trace estimation. Future work would involve proposing more efficient variants of score-matching that avoid drawing a large number of noise samples for computing the Hessian trace.

Finally, the score-matching strategy bears a striking resemblance to adversarial training methods. Having a low gradient norm in particular, has been identified as a important criterion to alleviate such adversarial examples, a property which it shares with score-matching. In addition, both methods result in improved gradient maps. However, one important distinction is that while adversarial training requires the Hessian norm to be small, score-matching naively requires minimization of the Hessian trace, which increases Hessian norm. Whether or not more stable variants of score-matching are in fact formally connected to adversarial training, is yet another topic for future research.

Bibliography

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. (2018). Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515.

Agarwal, R., Frosst, N., Zhang, X., Caruana, R., and Hinton, G. E. (2020). Neural additive models: Interpretable machine learning with neural nets.

Ancona, M., Ceolini, E., Oztireli, C., and Gross, M. (2018). Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations (ICLR 2018)*.

Avron, H. and Toledo, S. (2011). Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):1–34.

Ba, L. and Caruana, R. (2014). Do deep networks really need to be deep. *Advances in neural information processing systems*, 27:1–9.

Barocas, S., Hardt, M., and Narayanan, A. (2019). *Fairness and Machine Learning*. fairmlbook.org. http://www.fairmlbook.org.

Barrett, D. G. and Dherin, B. (2020). Implicit gradient regularization. *International Conference on Learning Representations*.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Bishop, C. M. (1995). Training with noise is equivalent to tikhonov regularization. *Neural Computation*.

Brendel, W. and Bethge, M. (2019). Approximating cnns with bag-of-local-features models works surprisingly well on imagenet.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Chalasani, P., Chen, J., Chowdhury, A. R., Jha, S., and Wu, X. (2018). Concise explanations of neural networks using adversarial training. *arXiv*, pages arXiv–1810.

Chang, C.-H., Creager, E., Goldenberg, A., and Duvenaud, D. (2019). Explaining image classifiers by adaptive dropout and generative in-filling.

Chen, C., Li, O., Tao, C., Barnett, A. J., Su, J., and Rudin, C. (2019). This looks like that: deep learning for interpretable image recognition.

Cho, J. H. and Hariharan, B. (2019). On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4794–4802.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863.

Cohen, N., Sharir, O., and Shashua, A. (2016). On the expressive power of deep learning: A tensor analysis. In Feldman, V., Rakhlin, A., and Shamir, O., editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 698–728, Columbia University, New York, New York, USA. PMLR.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Świrszcz, G., and Pascanu, R. (2017). Sobolev training for neural networks. *NIPS*.

Dombrowski, A.-K., Alber, M., Anders, C., Ackermann, M., Müller, K.-R., and Kessel, P. (2019). Explanations can be manipulated and geometry is to blame. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 13589–13600. Curran Associates, Inc.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv*.

Drucker, H. and Le Cun, Y. (1992). Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Tran, B., and Madry, A. (2019). Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*.

Erion, G., Janizek, J. D., Sturmfels, P., Lundberg, S., and Lee, S.-I. (2019). Learning explainable models using attribution priors.

Etmann, C., Lunz, S., Maass, P., and Schönlieb, C.-B. (2019). On the connection between adversarial robustness and saliency map interpretability. *arXiv preprint arXiv:1905.04172*.

Fong, R., Patrick, M., and Vedaldi, A. (2019). Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2950–2958.

Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR.

Furlanello, T., Zhao, J., Saxe, A. M., Itti, L., and Tjan, B. S. (2016). Active long term memory networks. *arXiv preprint arXiv:1606.02355*.

Ghorbani, A., Abid, A., and Zou, J. (2019). Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. (2020). Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*.

Haroush, M., Hubara, I., Hoffer, E., and Soudry, D. (2020). The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502.

Hastie, T. and Tibshirani, R. (1986). Generalized Additive Models. *Statistical Science*, 1(3):297 – 310.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heo, J., Joo, S., and Moon, T. (2019). Fooling neural network interpretations via adversarial model manipulation. In *Advances in Neural Information Processing Systems*, pages 2921–2932.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop*.

Hochreiter, S. and Schmidhuber, J. (1997). Flat minima. *Neural Computation*, 9(1):1–42.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. (2018). Evaluating feature importance estimates. *arXiv preprint arXiv:1806.10758*.

Hutchinson, M. F. (1990). A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709.

Jakubovitz, D. and Giryes, R. (2018). Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529.

Jung, H., Ju, J., Jung, M., and Kim, J. (2016). Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*.

Kaur, S., Cohen, J., and Lipton, Z. C. (2019). Are perceptually-aligned gradients a general property of robust classifiers? *arXiv preprint arXiv:1910.08640*.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. (2017). The (un) reliability of saliency methods. *arXiv preprint arXiv:1711.00867*.

Kindermans, P.-J., Schütt, K., Müller, K.-R., and Dähne, S. (2016). Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv preprint arXiv:1611.07270*.

Kingma, D. P. and LeCun, Y. (2010). Regularized estimation of image statistics by score matching. In *Advances in neural information processing systems*, pages 1126–1134.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

Leavitt, M. L. and Morcos, A. (2021). Selectivity considered harmful: evaluating the causal impact of class selectivity in dnns.

Li, O., Liu, H., Chen, C., and Rudin, C. (2018). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Li, Z. and Hoiem, D. (2016). Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer.

Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774.

Mahendran, A. and Vedaldi, A. (2016). Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255.

Micaelli, P. and Storkey, A. (2019). Zero-shot knowledge transfer via adversarial belief matching. *Advances in neural information processing systems*.

Mobahi, H., Farajtabar, M., and Bartlett, P. (2020). Self-distillation amplifies regularization in hilbert space. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3351–3361. Curran Associates, Inc.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932.

Mozer, M. C. and Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pages 107–115.

Nayak, G. K., Mopuri, K. R., Shaj, V., Radhakrishnan, V. B., and Chakraborty, A. (2019). Zero-shot knowledge distillation in deep networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4743–4751. PMLR.

Neal, R. M. et al. (2011). Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in neural information processing systems*, pages 3387–3395.

Nie, W., Zhang, Y., and Patel, A. (2018). A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. *arXiv preprint arXiv:1805.07039*.

Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., et al. (2018). Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Pang, T., Xu, T., Li, C., Song, Y., Ermon, S., and Zhu, J. (2020). Efficient learning of generative models via finite-difference score matching. *Advances in Neural Information Processing Systems*, 33.

Pearlmutter, B. A. (1994). Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160.

Phuong, M. and Lampert, C. (2019a). Towards understanding knowledge distillation. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5142–5151. PMLR.

Phuong, M. and Lampert, C. H. (2019b). Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1355–1364.

Quattoni, A. and Torralba, A. (2009). Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). On the expressive power of deep neural networks. *NIPS*.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Ross, A. S. and Doshi-Velez, F. (2017). Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *arXiv preprint arXiv:1711.09404*.

Ross, A. S., Hughes, M. C., and Doshi-Velez, F. (2017). Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*.

Rozantsev, A., Salzmann, M., and Fua, P. (2018). Beyond sharing weights for deep domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):801–814.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., and Zhong, C. (2021). Interpretable machine learning: Fundamental principles and 10 grand challenges. *arXiv preprint arXiv:2103.11251*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K.-R. (2016). Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673.

Santurkar, S., Ilyas, A., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Image synthesis with a single (robust) classifier. In *Advances in Neural Information Processing Systems*, pages 1260–1271.

Sau, B. B. and Balasubramanian, V. N. (2016). Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626. IEEE.

Shmelkov, K., Schmid, C., and Alahari, K. (2018). How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907.

Song, Y., Garg, S., Shi, J., and Ermon, S. (2019). Sliced score matching: A scalable approach to density and score estimation. *arXiv preprint arXiv:1905.07088*.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Srinivas, S. and Fleuret, F. (2018). Knowledge transfer with Jacobian matching. In *International Conference on Machine Learning*.

Srinivas, S. and Fleuret, F. (2019). Full-gradient representation for neural network visualization. In *Advances in Neural Information Processing Systems*, pages 4126–4135.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Subramanya, A., Pillai, V., and Pirsiavash, H. (2019). Fooling network interpretation in image classification. In *The IEEE International Conference on Computer Vision (ICCV)*.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Wang, S., Mohamed, A.-r., Caruana, R., Bilmes, J., Plilipose, M., Richardson, M., Geras, K., Urban, G., and Aslan, O. (2016). Analysis of deep neural networks with extended data jacobian matrix. In *International Conference on Machine Learning*, pages 718–726.

Weller, A. (2019). Transparency: motivations and challenges. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 23–40. Springer.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698.

Xu, H. and Mannor, S. (2012). Robustness and generalization. *Machine learning*, 86(3):391–423.

Yang, C., Xie, L., Su, C., and Yuille, A. L. (2019). Snapshot distillation: Teacher-student optimization in one generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2859–2868.

Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7130–7138. IEEE.

Yin, H., Molchanov, P., Alvarez, J. M., Li, Z., Mallya, A., Hoiem, D., Jha, N. K., and Kautz, J. (2020). Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724.

Yoo, J., Cho, M., Kim, T., and Kang, U. (2019). Knowledge extraction with no observable data. *Advances in neural information processing systems*.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *ICLR*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR.

Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., and Ma, K. (2019). Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722.

Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X., and Wang, T. (2020). Interpretable deep learning under fire. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*.

Zintgraf, L. M., Cohen, T. S., Adel, T., and Welling, M. (2017). Visualizing deep neural network decisions: Prediction difference analysis. *International Conference on Learning Representations*.

Suraj Srinivas

surajsrinivas.wordpress.com | suraj.srinivas@idiap.ch | suraj.srinivas@epfl.ch

Summary

I am a machine learning researcher, with active research interests in **interpretability** and **robustness** of machine learning models, and **compression** of deep neural networks.

Education

2017 - 2021 **Doctor of Philosophy**,
École Polytechnique Fédérale de Lausanne &
Idiap Research Institute, Switzerland,
Advisor: Prof. François Fleuret
Thesis: Gradient-based Methods for Deep Model Interpretability.

2014 - 2017 **Master of Science (Engineering)**,
Indian Institute of Science, Bangalore, India,
Advisor: Prof. R. Venkatesh Babu
Thesis: Learning Compact Architectures for Deep Neural Networks.

Work Experience

Aug 2020 - Jan 2021 **Research Intern**, *Qualcomm AI Research, Netherlands*,
Research on algorithms for improving neural network sparsity.

Jun-Aug 2016 **Research Intern**, *DataGrokr, India / Verisk Analytics, USA*, Speeding up inference on deep neural networks using tensor factorization.

Jan-Jun 2014 **Engineering Intern**, *Tonbo Imaging*, Bangalore,
Implemented image processing algorithms on FPGA for a thermal imaging camera.

Jun-Aug 2013 **Research Intern**, *Indian Institute of Science*, Bangalore,
Research on computational photography to perform camera jitter compensation.

Selected Publications

Google Scholar Profile

2021 **Suraj Srinivas** and François Fleuret. "Rethinking the Role of Gradient-based Attribution Methods in Model Interpretability", International Conference on Learning Representations (ICLR) **[Oral]**

2019 **Suraj Srinivas** and François Fleuret. "Full-Gradient Representation for Neural Network Visualization.", Neural Information Processing Systems (NeurIPS)
(PyTorch Implementation)

2018 **Suraj Srinivas** and François Fleuret. "Knowledge Transfer with Jacobian Matching.", International Conference on Machine Learning (ICML)

2017 **Suraj Srinivas** and François Fleuret. "Local Affine Approximations for Improving Knowledge Transfer.", NeurIPS Workshop on Learning with Limited Data **[Best Paper Award]**

2017 **Suraj Srinivas**, Akshayvarun Subramanya, R. Venkatesh Babu. "Training Sparse Neural Networks.", Computer Vision and Pattern Recognition Workshops (CVPRW)

2016 **Suraj Srinivas** and R. Venkatesh Babu. "Learning Neural Network Architectures using Backpropagation." British Machine Vision Conference (BMVC)

2015 **Suraj Srinivas** and R. Venkatesh Babu. "Data-free Parameter Pruning for Deep Neural Networks." British Machine Vision Conference (BMVC)

Talks

Apr 2021 Title: "**Rethinking the Role of Gradient-based Attribution Methods for Model Interpretability**"
Venue: ICLR (Virtual)

Jan 2020 Title: "**Neural Network Interpretability using Full-Gradient Representation**"
Venue: Indian Institute of Science, Bangalore

Jan 2020 Title: "**Full-Gradient Representation for Neural Network Visualization**"
Venue: ML for Astrophysicists Club (virtual)

Nov 2019 Title: "**Full-Gradient Representation for Neural Network Visualization**"
Event: Swiss Machine Learning Day, Lausanne

May 2019 Title: "**Complete Saliency Maps using Full-Jacobians**"
Event: Valais / Wallis AI workshop, Martigny

Jul 2018 Title: "**Knowledge Transfer with Jacobian Matching**"
Event: ICML, Stockholm

Jul 2016 Title: "**Making Deep Neural Networks Smaller and Faster**"
Event: Deep Learning Conf, Bangalore

Reviewing

Conferences AAAI, CVPR, ECCV, NeurIPS (2020) ; WACV, ICML, ICCV, NeurIPS (2021)

Journals IEEE SP-Letters, Elsevier Neural Networks, IEEE T-PAMI

Teaching

2018-2021 Teaching Assistant for Deep Learning Course (EE-559) at EFPL, Lausanne

Apr 2021 Guest Lecture on Interpretability for Deep Learning for Computer Vision Course (DS-265) at IISc, Bangalore

Miscellaneous

2014 Obtain rank **399** (out of $\sim$ 200k candidates) nation-wide in the Graduate Aptitude Test in Engineering for entrance to IITs / IISc for graduate studies in electronics and communications engineering

2012 Won first place in the E-Yantra nation-wide robotics contest held at IIT-Bombay, and was featured in The Times of India, New Indian Express and DH Education

2010 Obtain rank **191** (out of $\sim$ 100k candidates) state-wide in the Karnataka Common Entrance Test for entrance to state engineering colleges for undergraduate studies.