IDIAP RESEARCH REPORT

GRADIENT ALIGNMENT IN DEEP NEURAL NETWORKS

Suraj Srinivas [a] Francois Fleuret

Idiap-RR-14-2020

JULY 2020

[a] Idiap Research Institute

Gradient Alignment in Deep Neural Networks

Suraj Srinivas
Idiap Research Institute & EPFL
suraj.srinivas@idiap.ch

François Fleuret
Idiap Research Institute & EPFL
francois.fleuret@idiap.ch

Abstract

One cornerstone of interpretable deep learning is the high degree of visual alignment that input-gradients, *i.e.,* the gradients of the output w.r.t. inputs, exhibit with the input data. This alignment is assumed to arise as a result of the model's generalization, justifying its use for interpretability. However, recent work has shown that it is possible to 'fool' models into having arbitrary gradients while achieving good generalization, thus falsifying the assumption above. This leaves an open question: if not generalization, what causes input-gradients to align with input data?

In this work, we first show that it is simple to 'fool' input-gradients using the shift-invariance property of softmax, and that gradient structure is unrelated to model generalization. Second, we re-interpret the logits of standard classifiers as unnormalized log-densities of the data distribution, and find that we can improve this gradient alignment via a generative modelling objective called score-matching. To show this, we derive a novel approximation to the score-matching objective that eliminates the need for expensive Hessian computations, which may be of independent interest.

Our experiments help us identify one factor that causes input-gradient alignment in models, that being the approximate generative modelling behaviour of the normalized logit distributions.

1 Introduction

Input-gradients of trained deep neural networks, or gradients of outputs w.r.t. inputs, have been empirically observed to have a high degree of alignment with the inputs. For example, in image classification tasks, gradient magnitudes are observed to be higher on object locations and lower elsewhere. Folk wisdom states that these gradient magnitudes indicate the 'importance' placed by the model on different regions of the input, where larger gradient magnitudes indicate higher importance. This argument justifies their use as feature attribution maps for interpretation of discriminative models [1, 2, 3]. In this work, we show that input-gradients can be arbitrarily manipulated using the shift-invariance of softmax, which implies that input-gradient structure is unrelated to the discriminative capabilities of the model, thereby falsifying the assumption above.

Given that aligned input-gradients are not necessary for generalization, the reason for their emergence in standard deep models is puzzling. However from this observation, we can infer the presence of an *implicit regularizer* in neural network training that causes this gradient alignment. In this work, we wish to characterize this implicit regularizer, to understand the source of gradient alignment. For this purpose, we study the score-matching objective [4], which aims to align input-gradients with the gradients of the input data distribution, and is thus naturally formulated as a gradient alignment problem. To apply this, we exploit connections of discriminative classifiers with generative models [5, 6] by viewing the logits of standard classifiers as un-normalized log-densities. As the gradients of the input data distribution are unavailable, score-matching works by reducing the gradient alignment problem to that of local geometric regularization. Hence by combining these two techniques, the

Preprint. Under review.

generative modelling interpretation of logits and score-matching, we are able to connect the literature on generative models with that of geometric regularization of discriminative deep models.

In practice, the score-matching objective is known for being computationally expensive and unstable to train [7, 8], which has so far prevented its widespread usage for large-scale generative models. To this end, we also introduce approximations and regularizers which allow us to use score-matching on practical large-scale models. Aside from our usage in this paper, these methods may be of independent interest to the generative modelling community.

Overall, we make three contributions:

- We show in § 2 that it is trivial to fool input-gradients of standard classifiers using the shift-invariance of softmax, and that gradient alignment is unrelated to generalization.
- We devise in § 3 a tractable approximation to the score-matching objective that eliminates the need for expensive Hessian computations.
- We find in § 4 that improving generative modelling behaviour of discriminative models improves gradient alignment, and this helps us identify one possible reason for gradient-alignment in standard models, that being approximate generative modelling behaviour of the normalized logit distributions.

2 Fooling Gradients is Simple

Recently, it has been shown [9] that it is possible to train models into having arbitrarily structured input-gradients, while achieving good generalization. In this section, we show that it is trivial to 'fool' gradients of deep networks trained for classification, using the well-known shift-invariance property of softmax. Throughout the paper, 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.

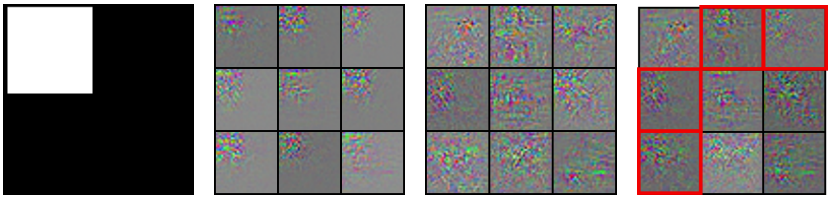
Observation. *Assume an arbitrary function $g : \mathbb{R}^D \to \mathbb{R}$. Consider another neural network function given by $f'_i(\cdot) = f_i(\cdot) + g(\cdot)$, for $0 \leq i \leq C$, for which we obtain $\nabla_{\mathbf{x}} f'_i(\cdot) = \nabla_{\mathbf{x}} f_i(\cdot) + \nabla_{\mathbf{x}} g(\cdot)$.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

We provide detailed arguments in the Supplementary material. 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 is unrelated to the discriminative capabilities of models. Despite this, a large fraction of work in interpretable deep learning [1, 10, 2, 11, 12] uses individual logits and logit-gradients for saliency map computation.

2.1 Fooling Loss-Gradients

Here, we show how we can also change loss-gradients arbitrarily without significantly changing the loss values themselves. In this case, we add slightly different scalar functions g_i to each logit.

Observation. *Assume an arbitrary function $g : \mathbb{R}^D \to \mathbb{R}^C$, such that $\max_{\mathbf{x}} |g_i(\mathbf{x}) - g_j(\mathbf{x})| \leq \epsilon$ for any two classes i, j. Consider another neural network function given by $f'_i(\cdot) = f_i(\cdot) + g_i(\cdot)$. For this, we have the following for small ϵ:*



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

Figure 1: 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.

$$
\begin{aligned}
|\ell_i'(\mathbf{x}) - \ell_i(\mathbf{x})| &\leq |\epsilon| + \mathcal{O}(\epsilon^2) \\
|\nabla_{\mathbf{x}}\ell_i'(\mathbf{x}) - \nabla_{\mathbf{x}}\ell_i(\mathbf{x})| &= \left|\sum_{j\neq i}\big(p_j(\nabla_{\mathbf{x}}g_j - \nabla_{\mathbf{x}}g_i) + (g_j - g_i)p_j(1-p_j)\nabla_{\mathbf{x}}f_j(\mathbf{x})\big)\right| + \mathcal{O}(\epsilon^2)
\end{aligned}
$$

Remark. *The error on the loss-gradients* $|\nabla_{\mathbf{x}}\ell_i'(\mathbf{x}) - \nabla_{\mathbf{x}}\ell_i(\mathbf{x})|$ *depends on both* $\nabla_{\mathbf{x}}f_j(\mathbf{x})$ *and* $(\nabla_{\mathbf{x}}g_j - \nabla_{\mathbf{x}}g_i)$*, whose magnitudes are unbounded, and thus can get arbitrarily large. E.g.: consider* g_i *being the zero function, and* g_j *being a high frequency sine wave with amplitude* ϵ*.*

Remark. *The approximation error for both the loss and loss-gradients are small for high-probability classes, and large for low-probability ones.*

The proof is provided in the supplementary material. Thus for inputs with low softmax probability, the loss-gradients can also be arbitrarily structured. Overall, the result above demonstrates that loss-gradients are also unreliable, as two models with very similar loss landscapes and hence discriminative abilities, can have drastically different loss-gradients.

2.2 Experiments

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 [9] consists of the following objective function.

$$\mathcal{L}_{fool}(\mathbf{s}, \mathbf{m}) = \mathbb{E}_{\mathbf{x}}\|\mathbf{s} - \mathbf{m}\|^2 \tag{1}$$

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 1a, 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 1b. 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 1d. This provides experimental validation for our theory.

3 Discriminative Classifiers as Generative Models

Our arguments in the previous section have demonstrated that we can easily cause input-gradients, particularly logit-gradients, to have arbitrary structure. In this section, we consider how to improve the alignment of the gradients with input data. To this end, we use the score-matching objective, is naturally formulated as a gradient alignment problem. For this, we first proceed by stating the link between generative models and the softmax function.

Let us 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 logits of 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.

3.1 Score-Matching

Score-matching [4] 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.

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{p_{data}(\mathbf{x})} \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})\|_2^2 && (2) \\
&= \mathbb{E}_{p_{data}(\mathbf{x})} \left(\text{trace}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})) + \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2 \right) + \texttt{const} && (3)
\end{aligned}
$$

The above relationship is proved [4] using integration by parts. This is a consistent objective, *i.e,* $J(\theta) = 0 \iff p_{data} = p_\theta$. Note that $\nabla_{\mathbf{x}} \log p_{data}(\mathbf{x})$ in equation 2 is unavailable, and thus equation 3 gets rid of this term. This 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 Hessians. However, equation 3 is intractable for high-dimensional data due to the Hessian trace term. To address this, we can use the Hutchinson's trace estimator [13] to efficiently compute an estimate of the trace by using random projections, which is given by:

$$
\text{trace}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})) = \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(0, \mathtt{I})} \, \boldsymbol{v}^\mathtt{T} \, \nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x}) \, \boldsymbol{v} \tag{4}
$$

This estimator has been previously applied to score-matching [14], and can be computed efficiently as this relies on Hessian-vector products, for which we can use Pearlmutter's trick [15]. However, this trick still requires two backward passes to compute a single Hessian-vector product, and in practice we may need to approximate the expectation using several Monte-Carlo samples. 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 $\sigma \in \mathbb{R}$.

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(0, \mathtt{I})} \, \boldsymbol{v}^\mathtt{T} \nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x}) \boldsymbol{v} &= \frac{1}{\sigma^2} \, \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(0, \sigma^2 \mathtt{I})} \boldsymbol{v}^\mathtt{T} \nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x}) \boldsymbol{v} \\
&\approx \frac{2}{\sigma^2} \, \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(0, \sigma^2 \mathtt{I})} \left(\log p_\theta(\mathbf{x} + \boldsymbol{v}) - \log p_\theta(\mathbf{x}) - \nabla_x \log p_\theta(\mathbf{x})^\mathtt{T} \boldsymbol{v} \right) \\
&= \frac{2}{\sigma^2} \, \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(0, \sigma^2 \mathtt{I})} \left(\log p_\theta(\mathbf{x} + \boldsymbol{v}) - \log p_\theta(\mathbf{x}) \right) && (5)
\end{aligned}
$$

Note that equation 11 involves a difference of log probabilities, which is independent of the partition function. For our case, $\log p_\theta(\mathbf{x} + \boldsymbol{v} | y = i) - \log p_\theta(\mathbf{x} | y = i) = f_i(\mathbf{x} + \boldsymbol{v}) - f_i(\mathbf{x})$. We have thus

considerably simplified and speeded-up the computation of the Hessian trace term, which now can be approximated **without** any backward passes, but using only a single additional forward pass. We present details regarding the variance of this estimator in the supplementary material.

3.2 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 2 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 [8] and noise conditional score networks [7] 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.

4 Experiments

In this section, we show that improving generative modelling of logit distributions leads to improved gradient alignment. For experiments, we shall consider the CIFAR100 dataset. Unless stated otherwise, the network structure we use shall be a 18-layer ResNet that achieves 77.12% accuracy on CIFAR100, and the optimizer used shall be SGD with momentum. 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, as shown in equation 6, with the stability regularizer discussed in §3.2. For this, we use a regularization constant $\lambda = 1e-3$. This model achieves 67.90% accuracy on the test set, which is a drop of about 10% compared to the original model.

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

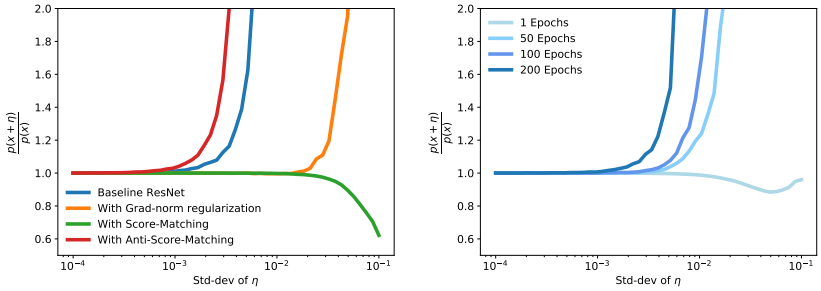
$$\underbrace{\ell_{reg}(f(\mathbf{x}),i)}_{\text{regularized loss}} = \underbrace{\ell(f(\mathbf{x}),i)}_{\text{cross-entropy}} + \lambda \left(\underbrace{\overbrace{h(\mathbf{x})}^{\text{Hessian-trace}} + \frac{1}{2}\overbrace{\|\nabla_{\mathbf{x}} f_i(\mathbf{x})\|_2^2}^{\text{gradient-norm}}}_{\text{score-matching}} + \underbrace{\overbrace{\mu}^{10^{-7}} h^2(\mathbf{x})}_{\text{stability regularizer}} \right) \tag{6}$$

Anti-score-matching We would like to have a tool that can *decrease* the score-matching tendency of a model, and thus possibly worsen the generative capabilites of models as a baseline. To enable this, we propose to increase 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 $\lambda = 1e-4$. Alternately, this can also be viewed as yet another logit-gradient 'fooling' method. This model achieves an accuracy of 76.56%.

Gradient-Norm regularization We observe that one the score-matching terms in equation 6 includes a gradient-norm regularizer. This has been used previously in the context of adversarial robustness [16], and as a regularizer in general. We hence propose to use this regularizer as another baseline for comparison, using a regularization constant of $\lambda = 1e-3$. This model achieves an accuracy of 76.26%.

4.1 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



(a) Density ratios of models trained with different regularizers

(b) Density ratios of a standard ResNet model across training epochs

Figure 2: Plots of density ratios representing local density profiles across varying levels of noise added to the input. **(a)** Most models with various regularizers assign higher densities to noisy inputs than clean ones, while the score-matched model is the only one that avoids this behaviour. **(b)** During training of a standard ResNet, models seem to get progressively worse at generative modelling, as models at later epochs assign high densities to inputs with smaller noise levels.

for gaussian noise η for different standard deviations, which are averaged across points in the entire dataset.

In Figure 2a, 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 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 regularization improves on this behaviour, but still assigns higher densities to inputs with large noise added. Finally, the score-matched model is the only one that assigns *lower* densities to noisy inputs than real inputs, which is the intended behaviour of a generative model. Thus we are able to obtain penalty terms that can both improve and deteriorate the generative modelling behaviour within discriminative models.

Our plots in figure 2b also indicate that standard models get progressively worse at generative modelling during training. *This indicates that the implicit regularizer responsible for gradient structure in standard models could be early stopping.* We examine this in more detail in §4.2.2.

4.2 Gradient Structure

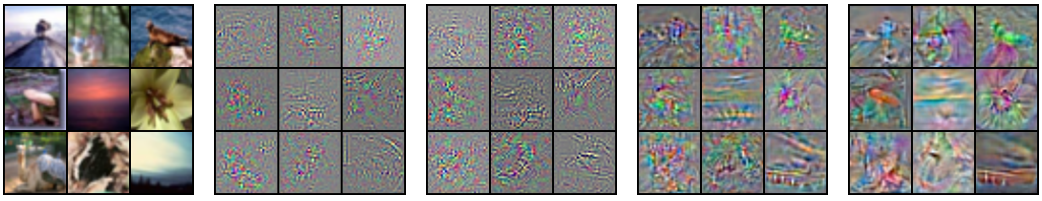
Here we visualize the structure of logit-gradients of different models as in Figure 3. We observe that gradient-norm regularized model and score-matched model have highly data aligned gradients, when compared to the baseline and anti-score-matched gradients. This shows that input-gradient alignment in neural networks can be significantly enhanced, and this is a function of the generative modelling behaviour of the logit distributions. In this visualization however, we do not see any discernable qualitative difference between the baseline and anti-score-matched gradients, nor are we able to make any quantitative statements about these. We hence propose to visualize and compare samples from these generative models.

4.2.1 Sampling from Model Distributions via Gradient Ascent

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 $\log p(\mathbf{x} \mid y = i) = f_i(\mathbf{x})$ starting from random noise input, which co-incidentally is a standard approach in interpretability [1]. The gradient ascent step is followed usually with a projection step to ensure that the resulting input lies in the range of valid image inputs, i.e.; between 0 and 1, and is given as follows:

$$\mathbf{x}_{t+1} \leftarrow \text{clamp}(\mathbf{x}_t + \alpha \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}|y = i), \text{min} = 0, \text{max} = 1)$$

Here, $\alpha \in R_+$ is the step size. Our results are shown in Figure 4. We notice that modes from the score-gradient trained model are significantly more realistic than baseline models. We also run a



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

Figure 3: Examples of logit-gradients for different models. While standard and anti-score-matched models **(a, b)** have minimally aligned gradients, gradients of models trained with gradient-norm regularization **(c)** exhibit some alignment, and score-matched models **(d)** exhibit the most alignment.

Model	**Sample Acc. (%)**	**Denoised Sample Acc. (%)**
Baseline ResNet	1.3	1.5
+ Anti-Score-Matching	1.3	1.3
+ Gradient Norm-regularization	32.7	37.8
+ Score-Matching	57.7	64.9

Table 1: Discriminative accuracy on VGG-11 of class-conditional samples generated from various ResNet-18 models. We observe that while the baseline and anti-score-matched models produce samples with close-to-random accuracies, the samples from gradient-norm regularized models and score-matched models achieve significantly better accuracies.

'denoising' experiment, where instead of starting with random noise, we start gradient ascent with data points perturbed with small noise. The modes of an ideal generative model lie near clean, un-noised data, thus motivating this experiment. Figure 5 shows that denoised samples of score-matched models are significantly more realistic than the rest.

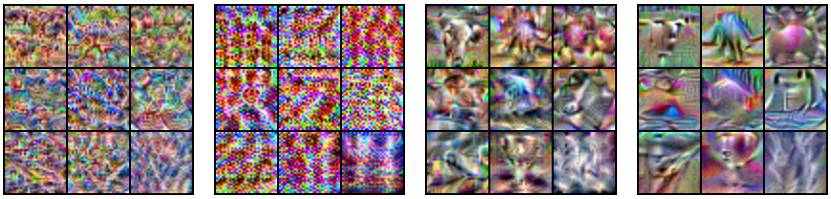
We also propose to measure quantitatively how well these generated samples adhere to the data distribution. In particular, we propose to measure the discriminative accuracy of these generated samples via a separately trained VGG-11 model. The intuition is that better class conditional generative images are more likely to be correctly classified irrespective of the model. In contrast with more popular metrics such as the inception-score, we would only like to capture sample realism and not diversity, thus motivating this measure. Like the inception score, this is also an approximate test. We show the results in table 1, which confirms the qualitative trend seen in samples above.

4.2.2 Effect of Early Stopping

Here we shall evaluate the effect of early stopping on generative modelling behaviour. For this, we train a standard ResNet model to 200 epochs, and plot the density ratios of models at 1, 50, 100 and 200 epochs in Figure 2b. We observe in Figure 6 as training progresses, the density ratios worsen over time. We also observe progressively worsening discriminative performance on these samples. While at epoch 25 we observed accuracies of (10.5 % , 14.8 %) respectively for raw and denoised samples, similar quantities for epoch 50 were (3.3 %, 4.7 %) and for epoch 150, (1.3 %, 1.3 %). This quantitatively indicates worsening generative modelling behaviour.

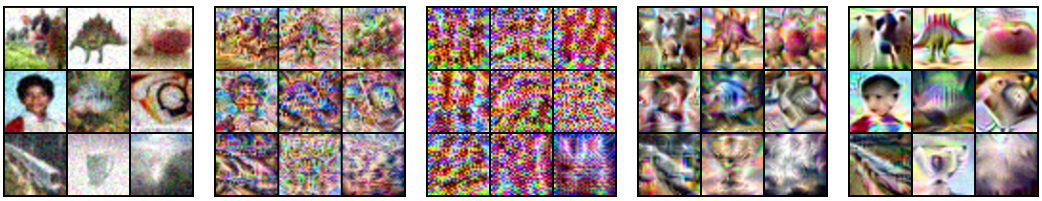
4.3 Properties of the Implicit Regularizer

Our experiments show that improving generative modelling behaviour, as evidenced by Figure 2a, leads to improved gradient alignment, as shown in Figures 3, 4, 5, and Table 1. This helps us identify one factor that can cause gradient alignment in standard models, that being approximate generative modelling of logit distributions. We also see that this generative modelling behaviour worsens during training, indicating that early stopping is one possible reason for this behaviour. To summarize, we find that early stopping may cause approximate generative modelling behaviour, which in turns causes gradient alignment. Score-matching also helps identify another factor related to approximate generative modelling, that being model smoothness, indicated by the gradient-norm regularization.



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

Figure 4: Samples generated from models by performing gradient ascent on random inputs. **(a)** Samples from the baseline model exhibit some noisy low-level structure, and **(b)** samples from anti-score-matched model are significantly noisier. **(c)** Sample quality is improved using gradient-norm regularization and **(d)** significantly so with score-matching.



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

Figure 5: 'Denoised' samples generated from models by performing gradient ascent on inputs perturbed with noise ($\sigma = 0.1$). Sample quality drastically improves with score-matching.

(a) Noisy Image (b) 25 epochs (c) 50 epochs (d) 150 Epochs

Figure 6: 'Denoised' samples generated from models at different epochs during training ($\sigma = 0.1$). We observe that sample quality progressively degrades across training epochs.

5 Conclusion

In this paper, we first found that input-gradients are not feature importance representations, and do not encode information regarding the discriminative capabilities of the model. Next, we found that improving the generative modelling behaviour of logit distributions lead to improved gradient alignment. This helped us identify approximate generative modelling as a cause of gradient alignment in standard models. To study this effect, we considered the score-matching approach and proposed scalable variants of the same.

However, in this paper we have only shown an empirical link between generative modelling and gradient alignment, and an analytical link is still missing. One hypothesis is that the statistics of natural images are responsible for such gradient alignment. Specifically, that the separation between a low-entropy 'object' which shows relatively little variation across images, and a high-entropy 'background' which virtually changes with every image, causes low-capacity generative models to treat the background regions as noise, thus suppressing their gradients. The resolution of this hypothesis would definitively resolve the gradient alignment paradox, and is left as an open problem.

References

[1] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[2] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[3] Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations (ICLR 2018)*, 2018.

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

[5] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020.

[6] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[7] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907, 2019.

[8] Durk P Kingma and Yann L Cun. Regularized estimation of image statistics by score matching. In *Advances in neural information processing systems*, pages 1126–1134, 2010.

[9] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. In *Advances in Neural Information Processing Systems*, pages 2921–2932, 2019.

[10] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626. IEEE, 2017.

[11] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[12] Suraj Srinivas and François Fleuret. Full-gradient representation for neural network visualization. In *Advances in Neural Information Processing Systems*, pages 4126–4135, 2019.

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

[14] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. *arXiv preprint arXiv:1905.07088*, 2019.

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

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

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

Supplementary Material

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.

$$\ell_i(\mathbf{x}) = -f_i(\mathbf{x}) + \log\left(\sum_{j=1}^{C}\exp(f_j(\mathbf{x}))\right) \tag{7}$$

$$\nabla_{\mathbf{x}}\ell_i(\mathbf{x}) = -\nabla_{\mathbf{x}}f_i(\mathbf{x}) + \sum_{j=1}^{C} p_j \nabla_{\mathbf{x}} f_j(\mathbf{x}) \tag{8}$$

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

Observation. *Assume an arbitrary function* $g : \mathbb{R}^D \to \mathbb{R}^C$*, such that* $\max_{\mathbf{x}} |g_i(\mathbf{x}) - g_j(\mathbf{x})| \leq \epsilon$ *for any two classes* i, j*. Consider another neural network function given by* $\tilde{f}_i(\cdot) = f_i(\cdot) + g_i(\cdot)$*. For this, we have the following for small* ϵ*:*

$$|\tilde{\ell}_i(\mathbf{x}) - \ell_i(\mathbf{x})| \leq |\epsilon| + \mathcal{O}(\epsilon^2) \tag{9}$$

$$|\nabla_{\mathbf{x}}\tilde{\ell}_i(\mathbf{x}) - \nabla_{\mathbf{x}}\ell_i(\mathbf{x})| = \left|\sum_{j\neq i}\big(p_j(\nabla_{\mathbf{x}}g_j - \nabla_{\mathbf{x}}g_i) + (g_j - g_i)p_j(1-p_j)\nabla_{\mathbf{x}}f_j\big)\right| + \mathcal{O}(\epsilon^2) \tag{10}$$

Proof. We start with equation 7, and write the expression for $\tilde{\ell}_i$.

$$\begin{aligned}
\tilde{\ell}_i(\mathbf{x}) &= -f_i(\mathbf{x}) - g_i(\mathbf{x}) + \log\left(\sum_{j=1}^{C}\exp(f_j(\mathbf{x}) + g_j(\mathbf{x}))\right) \\
&= -f_i(\mathbf{x}) + \log(\exp(-g_i(\mathbf{x}))) + \log\left(\sum_{j=1}^{C}\exp(f_j(\mathbf{x}) + g_j(\mathbf{x}))\right) \\
&= -f_i(\mathbf{x}) + \log\left(\sum_{j=1}^{C}\exp(f_j(\mathbf{x}) + g_j(\mathbf{x}) - g_i(\mathbf{x}))\right) \\
&= -f_i(\mathbf{x}) + \log\left(\sum_{j=1}^{C}\exp(f_j(\mathbf{x}))\exp(g_j(\mathbf{x}) - g_i(\mathbf{x}))\right) \\
&= -f_i(\mathbf{x}) + \log\left(\sum_{j=1}^{C}\exp(f_j(\mathbf{x}))(1 + g_j(\mathbf{x}) - g_i(\mathbf{x}))\right) + \mathcal{O}(\epsilon^2) \quad \to \text{Taylor Series} \\
&= -f_i(\mathbf{x}) + \log\left(\sum_{k=1}^{C}\exp(f_k(\mathbf{x}))\left(1 + \frac{\sum_{j=1}^{C}\exp(f_j(\mathbf{x}))(g_j(\mathbf{x}) - g_i(\mathbf{x}))}{\sum_{k=1}^{C}\exp(f_k(\mathbf{x}))}\right)\right) + \mathcal{O}(\epsilon^2) \\
&= \ell_i(\mathbf{x}) + \sum_{j=1}^{C} p_j(\mathbf{x})(g_j(\mathbf{x}) - g_i(\mathbf{x})) + \mathcal{O}(\epsilon^2) \quad \to \text{Taylor Series}
\end{aligned}$$

Upper bounding this expression using $\max_{\mathbf{x}} |g_i(\mathbf{x}) - g_j(\mathbf{x})| \leq \epsilon$, we obtain the equation 9. Differentiating this w.r.t. $\mathbf{x}$, we obtain equation 10.

□

Score-Matching Approximation

We consider the approximation derived for the estimator of the Hessian trace, which is first derived from Hutchinson's trace estimator [13]. 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.

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,\mathtt{I})}\, \boldsymbol{v}^\mathtt{T}\nabla^2_\mathbf{x} f(\mathbf{x})\boldsymbol{v} &= \frac{1}{\sigma^2}\,\mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,\sigma^2\mathtt{I})}\boldsymbol{v}^\mathtt{T}\nabla^2_\mathbf{x} f(\mathbf{x})\boldsymbol{v} \\
&= \frac{2}{\sigma^2}\,\mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,\sigma^2\mathtt{I})}\left(f(\mathbf{x}+\boldsymbol{v}) - f(\mathbf{x}) - \nabla_x f(\mathbf{x})^\mathtt{T}\boldsymbol{v} + \mathcal{O}(\sigma^3)\right) \\
&= \frac{2}{\sigma^2}\,\mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,\sigma^2\mathtt{I})}\left(f(\mathbf{x}+\boldsymbol{v}) - f(\mathbf{x})\right) + \mathcal{O}(\sigma) \qquad (11)
\end{aligned}$$

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*.

$$\text{Taylor Trace Estimator (TTE)} = \frac{2}{N\sigma^2}\,\sum_{i=1}^{N}\big(f(\mathbf{x}+\boldsymbol{v}_i) - f(\mathbf{x})\big) \quad \text{s.t.} \quad \boldsymbol{v}_i \sim \mathcal{N}(0,\sigma^2\mathtt{I}) \qquad (12)$$

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.

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

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:

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

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 [4], 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 [13, 17]:

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