Knowledge Transfer with Jacobian Matching

Suraj Srinivas [1] **François Fleuret** [1]

Abstract

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 paper, 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.

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 & 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 paper 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 Jacobians, and by Zagoruyko & Komodakis (2017) who viewed Jacobians as attention maps. However it was unclear how these methods were related to classical distillation approaches (Ba & Caruana, 2014; Hinton et al., 2015), making it difficult to identify reasons for improved performance.

Recently Li & 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 our paper 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 & 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 Jacobian-norm penalties can be used to learn models robust to noise.

2. Related Work

Several Jacobian-based regularizers have been proposed in recent times. Sobolev training (Czarnecki et al., 2017), showed that using higher order derivatives along with the targets can help in training with less data. This work is similar

[1]Idiap Research Institute & EPFL, Switzerland. Correspondence to: Suraj Srinivas <suraj.srinivas@idiap.ch>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

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. Similarly, Wang et al. (2016) used Jacobians for distillation and showed that it helps improve performance. Zagoruyko & Komodakis (2017) introduced the idea of matching attention maps, of which Jacobians were an instance. This work found that combining both activation matching and Jacobian matching was helpful, which is a natural consequence of analysis in our work.

Drucker & Le Cun (1992) considered penalizing the Jacobian 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 & 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 & Komodakis (2017) showed how to enhance distillation by matching intermediate features along with the outputs, but used different methods to do so. Sau & Balasubramanian (2016) found that adding noise to logits helps during teacher-student training. We show that the use of the Jacobian can be interpreted as adding such noise to the inputs analytically.

3. Jacobians of Neural Networks

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

$$f(\mathbf{x} + \Delta\mathbf{x}) = f(\mathbf{x}) + \nabla_x f(\mathbf{x})^T(\Delta\mathbf{x}) + \mathcal{O}(\epsilon^2) \quad (1)$$

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.

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 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.

3.2. 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 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?

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 & 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 & 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*

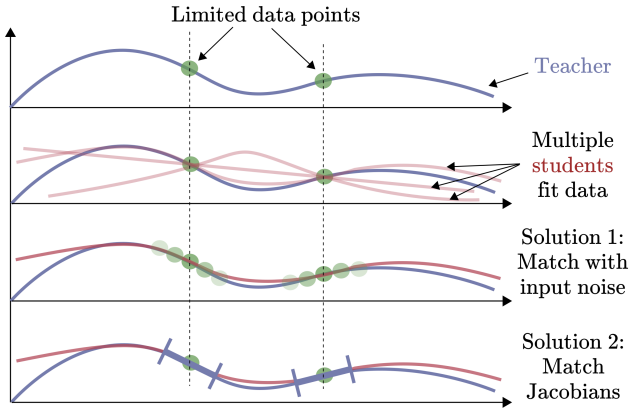

Figure 1. Illustration of teacher-student learning in a simple 1D case. Here, x-axis is the input data, and y-axis denotes function outputs. Given a limited number of data points, there exist multiple student functions consistent with the data. How do we select the hypothesis closest to the teacher's? 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.

variable $\mathbf{z} \in \mathbb{R}^D$ *with scaling factor* $\sigma \in \mathbb{R}$*. Then,*

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

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:

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

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 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.*

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

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 & 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 Jacobian rather than weights. We can derive a similar result for the case of cross-entropy error as well, which is given by -

$$\mathbb{E}_{\boldsymbol{\xi}}\left[-\sum_{i=1}^{k}y^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x}+\boldsymbol{\xi}))\right] \approx -\sum_{i=1}^{k}y^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x})) \;+\; \sigma^2\sum_{i=1}^{k}y^i(\mathbf{x})\frac{\|\nabla_x\mathcal{S}_s^i(\mathbf{x})\|_2^2}{\mathcal{S}_s^i(\mathbf{x})^2} \tag{3}$$

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 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.

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 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 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).

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

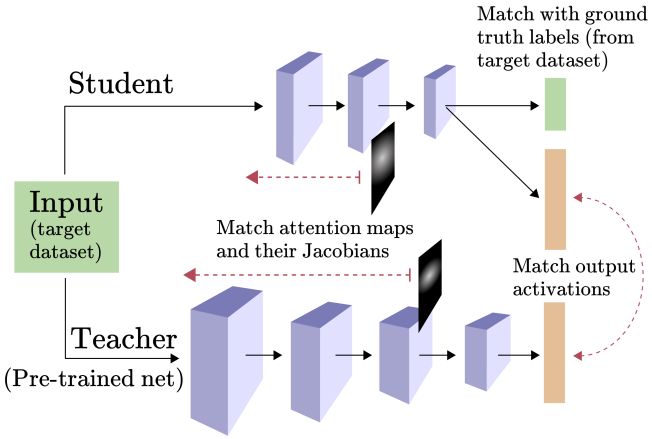

Figure 2. Illustration of our proposed method for transfer learning. We match the output activations of a pre-trained Imagenet network similar to LwF (Li & Hoiem, 2016). We also match aggregated activations or "attention" maps between networks, similar to the work of Zagoruyko & Komodakis (2017). We propose to match Jacobians of (aggregated) attention maps w.r.t. inputs.

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 & Hoiem (2016), called "Learning without Forgetting" (**LwF**). Note that similar 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. 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 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.

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

$$\|\rho(\mathbf{x}_1) - \rho(\mathbf{x}_2)\| \leq \mathrm{K}\psi_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2) \tag{4}$$

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:

$$\mathcal{H}_a(A, B) = \sup_{a \in A} \inf_{b \in B} \psi_{\mathbf{x}}(a, b). \tag{5}$$

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*

$$\frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{x} \sim \mathcal{D}_s} \ell(f(\mathbf{x}), g(\mathbf{x})) \leq \max_{\mathbf{x} \sim \mathcal{D}_t} \ell(f(\mathbf{x}), g(\mathbf{x})) \tag{6}$$

$$+ \mathrm{K}\mathcal{H}_a(\mathcal{D}_s, \mathcal{D}_t) \tag{7}$$

On the left side of 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 (7). 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 & 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.

5.1.1. Incorporating Jacobian matching

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.

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 & 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 -

$$A = AttMap(Z) = \sum_{i \in channels} |Z_i|^2 \tag{8}$$

Further, the loss function used to match these activations is

$$\text{Match Activations} = \left|\left| \frac{A_t}{\|A_t\|_2} - \frac{A_s}{\|A_s\|_2} \right|\right|_2 \tag{9}$$

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

5.2.1. 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 9, given by

$$\text{Match Jacobians} = \left|\left| \frac{\nabla_x f(\mathbf{x})}{\|\nabla_x f(\mathbf{x})\|_2} - \frac{\nabla_x g(\mathbf{x})}{\|\nabla_x g(\mathbf{x})\|_2} \right|\right|_2^2 \tag{10}$$

We present a justification for this in the supplementary material.

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 & Hinton, 2009). Second, we show on that same dataset that penalizing Jacobian norm increases stability of networks to random noise. Finally, we perform transfer learning experiments on the MIT Scenes dataset (Quattoni & Torralba, 2009). We provide more detail about the experimental setups in the supplementary material.

6.1. Distillation

For the distillation experiments, we use VGG-like (Simonyan & 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 1 we can conclude that (1) it is generally beneficial to do any form of 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 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.

6.2. Noise robustness

We perform experiments where we penalize the Jacobian 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 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.

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 & 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 & 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 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.

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.

6.3.1. 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 & 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 & 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 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.

6.3.2. 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 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 $(\text{feature size})/5$ 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.

Table 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 $\rightarrow$	1	5	10	50	100	500 (full)
Cross-Entropy (CE) training	5.69	13.9	20.03	37.6	44.92	54.28
CE + match activations	12.13	26.97	33.92	46.47	50.92	**56.65**
CE + match Jacobians	6.78	23.94	32.03	45.71	51.47	53.44
CE + match {activations + Jacobians}	**13.78**	**33.39**	**39.55**	**49.49**	**52.43**	54.57
Match activations only	10.73	28.56	33.6	45.73	50.15	56.59
Match {activations + Jacobians}	13.09	33.31	38.16	47.79	50.06	51.33

Table 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 Jacobian-norm penalty regularizer. γ is the ℓ_2 regularization strength and p is the dropout value. We see that the Jacobian-norm penalty can be remarkably robust to noise, unlike ℓ_2 regularization and dropout.

Noise std. dev. $\rightarrow$	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
$\lambda = 0.1$	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 regularization ($\gamma = 1e-3$)	**66.92**	60.41 ± 0.27	39.93 ± 0.28	23.32 ± 0.25	13.76 ± 0.16
Dropout ($p = 0.25$)	66.8	61.53 ± 0.10	44.34 ± 0.19	26.70 ± 0.24	15.77 ± 0.11

Table 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. Results are averaged over two runs.

# of Data points per class $\rightarrow$	5	10	25	50	Full
Cross-Entropy (CE) training on untrained student network	11.64	20.30	35.19	46.38	59.33
CE on pre-trained student network (Oracle)	**25.93**	**43.81**	**57.65**	**64.18**	**71.42**
CE + match activations (Li & Hoiem, 2016)	17.08	27.13	45.08	55.22	65.22
CE + match {activations + Jacobians}	17.88	28.25	45.26	56.49	66.04
CE + match {activations + attention} (Zagoruyko & Komodakis, 2017)	16.53	28.35	46.01	57.80	**67.24**
CE + match {activations + attention + Jacobians}	**18.02**	**29.25**	**47.31**	**58.35**	**67.31**

Table 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
Jacobian loss reduction (%)	**25.88**	15.59	11.55	1.25

Table 5. Ablation experiments over the computation of Jacobian. 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

7. Conclusion

In this paper 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.

Acknowledgements

This work was supported by the Swiss National Science Foundation under the ISUL grant FNS-30209.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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