Tractable Neural Networks for Identity Recognition

David Eng
dkeng@stanford.edu

Jenny Hong
jyunhong@stanford.edu

Abstract

Deep neural networks are state-of-the-art for many applications, but the weights that parametrize them require significant computational resources, making them inaccessible to less powerful devices (eg. mobile). In this project, we aim to reduce the computational demand of existing approaches for identity recognition. We propose the use of augmented hints *which extend on the idea of hints from the FitNets model described by Romero et al., combining it with knowledge distillation as proposed by Hinton and Dean. Augmented hints use the outputs and intermediate representations of a larger teacher network to improve the training process of a smaller student network. The two-step augmented hints process first initializes weights up to an intermediate layer of the student, using an intermediate layer of the teacher. The second step involves training the full student network, using the teacher network to regularize. Using augmented hints definitely allows student networks to achieve higher accuracies with fewer training epochs. Our results show that the larger the student network, the more the student accuracy improves as a result of using the augmented hints method. However, our results do not support the idea that augmented hints reduces the time it takes a small model to get to a fixed accuracy. With very simple models, there is not enough complexity in the network for the time it takes to find a good initialization to pay off at training or test time.*

1. Introduction

1.1. Problem

Deep neural networks have set numerous records in a number of computer vision and other computational applications. However, the most effective neural networks used for visual recognition tasks require gigabytes of memory to store and weeks of time to compute. These significant computational loads make it difficult to use deep neural networks to solve computer vision tasks specific to mobile devices and other devices ordinary consumers have access to, where computational resources are limited. Existing methods for individuals to leverage neural networks for personal data involve interfacing with an external server. However, this approach (1) requires Internet connection for instantaneous use and (2) sacrifices users' privacies by communicating personal data and model parameters to an external service.

A mobile-specific computer vision task of interest to us is identity recognition, applied to image retrieval. For example, mobile users may want to query their devices for "all selfies with Alice and Bob." We could complete these queries in a remote data center; however, we imagine that users might want this feature on their phones in situations without a reliable network connection. Considering this use case, we will design a identity recognition system that functions without the cloud and limited computing power.

1.2. Approach

In this paper, we study the tradeoffs between accuracy and computational tractability as it applies to identity recognition. In particular, we extend the idea of the FitNets model described by Romero et al. [6], an example of network compression via student-teacher networks. The student-teacher networks operate by training a smaller student network to imitate a larger teacher network or ensemble of networks. Before training, an intermediate representation of the teacher network is used to train the weights of the layers up to some (smaller) intermediate layer of the student network. We also borrow ideas from knowledge distillation as proposed by Hinton and Dean. [3]. During training, the final representation of the larger teacher network is used in regularizing the learning of the student network. Our specific models and loss functions are described in Section 3, and results are presented in Section 4.

2. Background

2.1. Identity Recognition

One existing deep neural network facial recognition architecture is the Facenet [7] architecture. The FaceNet model, rather than directly classifying a face into one of a fixed number of personal identity classes, produces an *embedding* in a 128-dimensional unit hypersphere, with the property that a larger distance between two face embed-

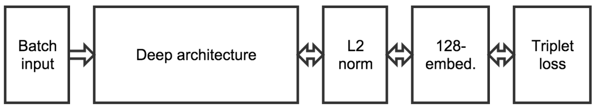


Figure 1: FaceNet model structure

dings means that the faces are likely not of the same person. Figure 1 shows the general structure used in training.

We detail a particular FaceNet architecture in Section 3, and the others are similar but of different sizes. Here, we describe the loss function used in the particular application of identity recognition, known as *triplet embedding loss*. A triplet indexed by i is defined as (x_i^a, x_i^p, x_i^n), where x_i^a is an *anchor* image of a specific person, x_i^p is a *positive* image of the same person, and x_i^n is a *negative* image of some other person. Let $f(x)$ be the embedding of image x. The loss $\mathcal{L}_{TE}(f)$ is then given by

$$\sum_i^N \max\{0, \|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha\},$$

where N is the number of all possible triplets and α is a parameter for the margin enforced between positive and negative pairs. In essence, the loss penalizes any triplet where the negative image is not more than α farther than the positive image from the anchor image.

One open source implementation based on the FaceNet model is OpenFace [1], which uses Python and Torch [2].

2.2. Knowledge Distillation

The knowledge distillation framework proposed by Hinton and Dean [3] trains a student network from a teacher network, which can be a single network or an ensemble of networks. If the teacher is an ensemble, we assume that the averaging required to combine their outputs into a single output has been done before this framework. Let a_T and a_S represent the pre-softmax outputs of the teacher and student networks, respectively. Knowledge distillation defines softened outputs $P_T^\tau = \sigma(\frac{a_T}{\tau}), P_S^\tau = \sigma(\frac{a_s}{\tau})$ of the teacher and student, respectively, for a parameter $\tau > 1$. The student network is trained to minimize the following loss function:

$$\mathcal{L}_{KD}(W_S) = \mathcal{H}(y_{\text{true}}, P_S) + \lambda \mathcal{H}(P_T^\tau, P_S^\tau),$$

where W_S are the weights of the student network, P_S is the student's outputs with softmax, $\mathcal{H}$ is cross-entropy, and λ is a regularization that enforces the student network to learn more from the teacher network than from only the labels.

2.3. Hint-Based Training

Hint-based training as proposed by Romero et al. [6] again uses the student-teacher framework to train student networks. Hint-based training uses a "guided" layer G in the student network and a "hint" layer H in the teacher network. Layers 1 through G of the student layer are trained to minimize the following loss function:

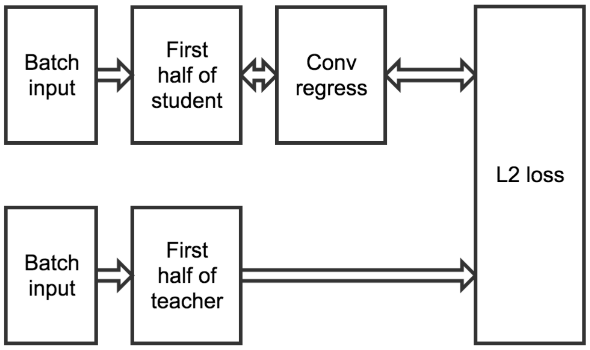


Figure 2: Hint-Based Training

$$\mathcal{L}_{HT}(W_G, W_r) = \frac{1}{2}\|u_h(x; W_H) - r(v_g(x; W_G); W_r)\|_2^2,$$

where u_h and v_g are the teacher/student respresentations at their respective hint/guided layers with parameters W_H and W_G, r is the regressor function on top of the guided layer so that the dimensions match the outputs of u_h, with parameters W_r.

The particular model described is known as FitNets, because these models are, compared to their teacher networks, "thin" (fewer parameters at each layer) and "tall" (more layers). The FitNet model uses two stages. It first uses hint-based training, and then in sequence uses knowledge distillation training.

3. Methods

3.1. Augmented Hints

We implement a new framework known as *augmented hints*, combining knowledge distillation and hint-based training in the specific application of producing facial embeddings.

The first step of the augmented hints framework is to use hint-based training to initialize the student network. A schematic of the process is shown in Figure 2. Double arrows denote links followed in both forward and backward passes, and forward arrows denote only a forward pass. We are given a teacher model whose weights have already been trained. A fixed layer H in the teacher network and a fixed layer G in the student network are chosen, and layers 1 through G of the student network are trained to minimize $\mathcal{L}_{HT}(W_G, W_r)$, where W_r are the weights of a convolutional regressor.

Figure 2 shows that this is done by computing a forward pass in the teacher up to layer H, computing a forward

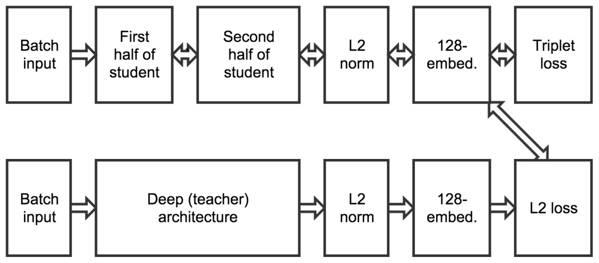


Figure 3: Augmented hints framework

pass in the student up to layer G, passing the student representation through a regressor, and then computing the loss between the teacher output and the regressed student output. The gradients from this loss are then backpropagated through only the student, and through only the first G layers of the student. After just a relatively small number of epochs, this procedure produces a strong initialization for the student network.

The next step is to incorporate a form of knowledge distillation to aid in training the full student network. This is a loss based on the 128-dimensional embeddings of the student and the teacher E_S and E_T, respectively. We introduce a new loss function:

$$\mathcal{L}_{TH}(W_S) = (1-\lambda)\mathcal{L}_{TE}(E_S) + \lambda\|E_S - E_T\|_2^2,$$

where $\mathcal{L}_{TE}$ is the triplet embedding loss introduced in Seciton 2.1 as used in the FaceNet framework and λ is a regularization that enforces the student network to learn from the teacher network.

Figure 3 shows the training pipeline. In parallel, the student and teacher each perform a forward propagation. (Note that the convolutional regressor from the hints-based training for initialization is NOT part of the student network.) The output of both the student and the teacher are fed into the $L2$ loss, which refers to the $\|E_S - E_T\|_2^2$ term of $\mathcal{L}_{TH}(W_S)$. The output of the student is also fed into the triplet loss, which corresponds to the $\mathcal{L}_{TE}(E_S)$ term of $\mathcal{L}_{TH}(W_S)$. These two loss functions both produce gradients, and a weighted sum of these gradients is fed back into the output (embedding) layer of the student network and used in backpropagation.

3.2. Teacher Network

For the teacher network, we use the NN4 Small network in OpenFace, which is a smaller version of the NN2 Inception architecture in the FaceNet system [7]. This architecture is depicted in Figure 5. The size of the conv and pool operations are given in the format (F, S, P), where F is the size of the filter, S is the step size, and P is the padding, in each of width and height. The Inception layers [8] are the same as those described in the original FaceNet system, except that the (5a) and (5b) layers do not have the 5×5 kernels and produce smaller output layers. The output of the last layer is taken to be 128 by default.

type	output size
conv (7, 2, 3)	$48 \times 48 \times 64$
maxpool (3, 2, 1) and norm	$24 \times 24 \times 64$
maxpool (3, 2, 1) and norm	$24 \times 24 \times 64$
inception (2)	$24 \times 24 \times 192$
maxpool (3, 2, 1) and norm	$12 \times 12 \times 192$
inception (3a)	$12 \times 12 \times 256$
inception (3b)	$12 \times 12 \times 320$
inception (3c)	$6 \times 6 \times 640$
inception (4a)	$6 \times 6 \times 640$
inception (4e)	$3 \times 3 \times 1024$
inception (5a)	$3 \times 3 \times 736$
inception (5b)	$3 \times 3 \times 736$
average pool (3, 1, 0)	$1 \times 1 \times 736$
fully connected	$1 \times 1 \times 128$
L2 normalization	$1 \times 1 \times 128$

Figure 4: Small NN4 Architecture

3.3. Implementation Details

We implemented our augmented hints method as a new branch in our own fork of the OpenFace project. Therefore, we integrated our model definitions and training methods with their existing pipeline. OpenFace provides pre-trained models for their NN4 Small (version 2) network.

Because we wanted our code to fit within the framework of the OpenFace pipeline, we did not introduce entire new classes for the augmented hints framework. Rather, we added methods and options in the existing framework to run through the training procedure twice: first for part of the student network with an extra convolutional layer, and second for the entire student network, copying over the weights from initialization.

The convolutional layer was chosen in the following way. We choose the layer G in the student network such that its the same spatial size as the output of layer H in the teacher network. Only the depths differ. We implemented the convolutional regressor by adding a spatial convolutional layer of $1 \times 1 \times D_T$ filters, where D_T is the depth of the teacher network at layer H. This regresses the output of the student network into an output of the same size as the teacher network, which we can then compare using the $L2$ norm.

4. Experiment

4.1. Student Architectures

We experimented with a number of student architectures. To control the number of variables in the experiment, all of the students whose results are reported used a structured architecture. In particular, the student networks had 9 iterations of the spatial convolution, batch normalization, and ReLU. This was followed by an average pooling layer to reduce the spatial size to 1×1. The last two layers were a fully connected layer, followed by an $L2$ normalization. The guided layer G used in hint-based training was always the output of the 6th ReLU.

The variables we restricted ourselves to were the sizes of the filters and the number of filters in each spatial convolutional layer. This makes it easy to compare pairs of networks that differ only in the sizes of the filters used, or to compare networks that differ only in the number of filters, or to compare networks by number of parameters.

	Model 5	Model 6	Model 7
conv1	(5, 2, 2)	(5, 2, 2)	(5, 2, 2)
out1	$48 \times 48 \times 8$	$48 \times 48 \times 8$	$48 \times 48 \times 8$
conv2	(5, 2, 2)	(5, 2, 2)	(5, 2, 2)
out2	$24 \times 24 \times 16$	$24 \times 24 \times 16$	$24 \times 24 \times 10$
conv3	(5, 1, 2)	(5, 1, 2)	(5, 1, 2)
out3	$24 \times 24 \times 24$	$24 \times 24 \times 24$	$24 \times 24 \times 12$
conv4	(5, 2, 2)	(5, 2, 2)	(5, 2, 2)
out4	$12 \times 12 \times 28$	$12 \times 12 \times 28$	$12 \times 12 \times 16$
conv5	(3, 2, 1)	(3, 2, 1)	(3, 2, 1)
out5	$6 \times 6 \times 32$	$6 \times 6 \times 32$	$6 \times 6 \times 20$
conv6	(3, 1, 1)	(3, 1, 1)	(3, 1, 1)
out6	$6 \times 6 \times 36$	$6 \times 6 \times 36$	$6 \times 6 \times 24$
conv7	(3, 2, 1)	(3, 2, 1)	(3, 2, 1)
out7	$3 \times 3 \times 48$	$3 \times 3 \times 40$	$3 \times 3 \times 28$
conv8	(3, 1, 1)	(3, 1, 1)	(3, 1, 1)
out8	$3 \times 3 \times 64$	$3 \times 3 \times 44$	$3 \times 3 \times 32$
conv9	(3, 1, 1)	(3, 1, 1)	(3, 1, 1)
out9	$3 \times 3 \times 128$	$3 \times 3 \times 48$	$3 \times 3 \times 36$
# param	183224	103540	47374

Figure 5: Sample student architectures

Figure 5 shows the convolutional filter sizes and intermediate representation sizes for each of the convolutional layers in some example student models (which we index 5, 6, 7).

Note that these are much smaller than the original FaceNet NN4 Small2 model, which contains around 5 million parameters. Model 5 is a mere 3% of the original model! Model 6 is around 2%, and Model 7 more than 100 times smaller than even the smaller version of FaceNet NN4.

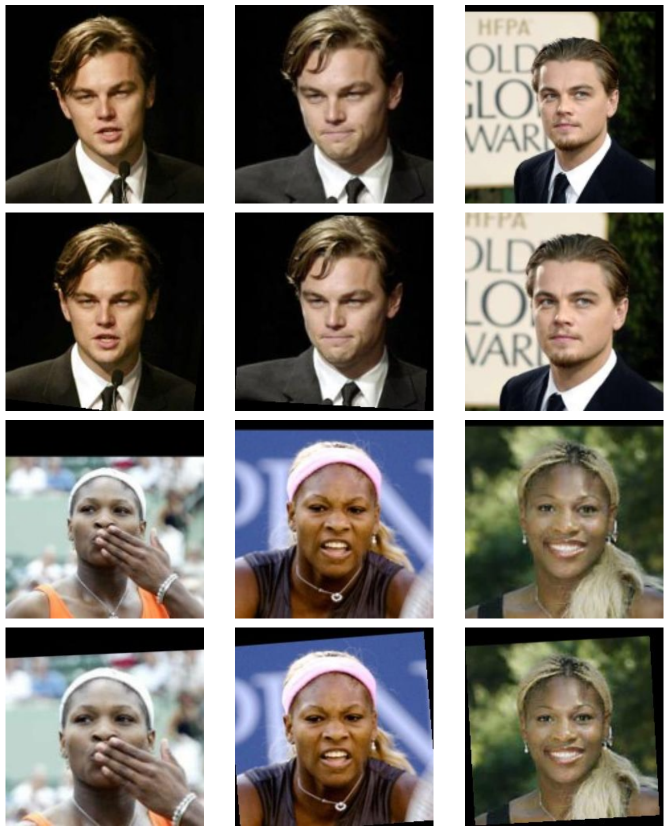


Figure 6: Raw (upper) and deep funneled (lower) images from Labeled Faces in the Wild

4.2. Dataset

We use the Labeled Faces in the Wild [5] dataset from the University of Massachussetts, Amherst, which is a dataset of faces detected by the Viola-Jones framework and labeled with their identities. It contains 13233 images of 5749 distinct people. We filter this dataset to use only those people who have at least some number of instances. 1680 people have 2 or more images, and 898 people have 3 or more images. We use a preprocessing step called deep funnelling [4] to align images. Figure 6 shows images of raw and deep funneled images from Labeled Faces in the Wild. OpenFace [1] reports that the aligning process can account for significant parts of the time their system requires to produce an embedding of an image, up to 20 or 30 percent.

4.3. Metrics

We are interested in the tradeoff between computational tracatbility and accuracy. Because we do not have a fixed number of classes, accuracy is not measured by traditional classification accuracy. Rather, we use nearest neighbor classification on the embeddings of a test image and the training images. Accuracy is evaluated according to pairs of images. The Labeled Faces in the Wild dataset uses a ground truth list of positive pairs of images (images that are of the same person) and list of negative pairs of images (images that are not of the same person). Our classification accuracy is the classification accuracy with respect to this list.

We are interested in the effect of a teacher network on a student network in aiding the student's training process. We compare the accuracy of a student network architecture trained with no hints or knowledge distillation compared to the accuracy of a student network trained using augmented hints. After every epoch, we save a the state of the two networks and evaluate the accuracy on each network for each epoch.

In all experiments, we split our dataset into training and validation on the image level. That is, a single person may have some of his or her photos in the training dataset and the others in the validation dataset.

Then, we evaluate the accuracy of our learned embedding by having it predict whether various pairs of images in the dataset contains the same or different persons.

5. Results

	time	train acc.	num. params
nn4.small2.v1.t4	1303	0.8525	3733968
Model 5	43.5	0.8380	183224
Model 6	35.2	0.8368	103540
Model 7	33.6	0.8230	47374

Figure 7: Validation Accuracy, Time, and Space Tradeoffs

All our results should be viewed from the perspective of a computationally starved environment. The student networks range in size from 1% to 5% of the teacher network (FaceNet NN4 Small). We train these student networks for up to 10 epochs each, while the default setting in the OpenFace implementation is to train for 1000 epochs, for the same batch size.

Figure 7 shows the time required to train 10 epochs, the training accuracy at 10 epochs, and the total number of parameters for the teacher network and our sample student networks.

At face value, we significantly reduce training time and space required to store the models for little loss in training accuracy at 10 epochs. However, we conduct the following experiments to determine whether we can leverage augmented hints to further reduce the time required to train one of these smaller networks.

5.1. Varying Hint Training Epochs

We conducted one experiment by varying the number of epochs spent on the first step of our procedure. Recall that the first step of augmented hints uses hint-based training to train the first G layers of the student network. The variable we use is the number of epochs spent on this part of the training process.

Figure 8 shows the results of this experiment on models 5, 6, and 7. Recall that the second step of augmented hints uses a system similar to knowledge distillation to train the full student network.

The horizontal axis of each of the nine plots refers to the number of epochs in this part of the training. The green line shows the validation accuracy of some model for some variable setting. For each model, we also include the performance of a control model in a dotted black line. The dotted black line shows the performance of the accuracy for the corresponding architecture trained versus number of epochs trained for under vanilla backpropagation.

From this figure we see that for larger models (e.g. model 5), initializing using augmented hints allows the model to achieve much higher accuracy in many fewer epochs. However, we found that when the model became very small, with only a dozen or two filters per level, (e.g. model 7), augmented hints does not seem to improve the accuracy.

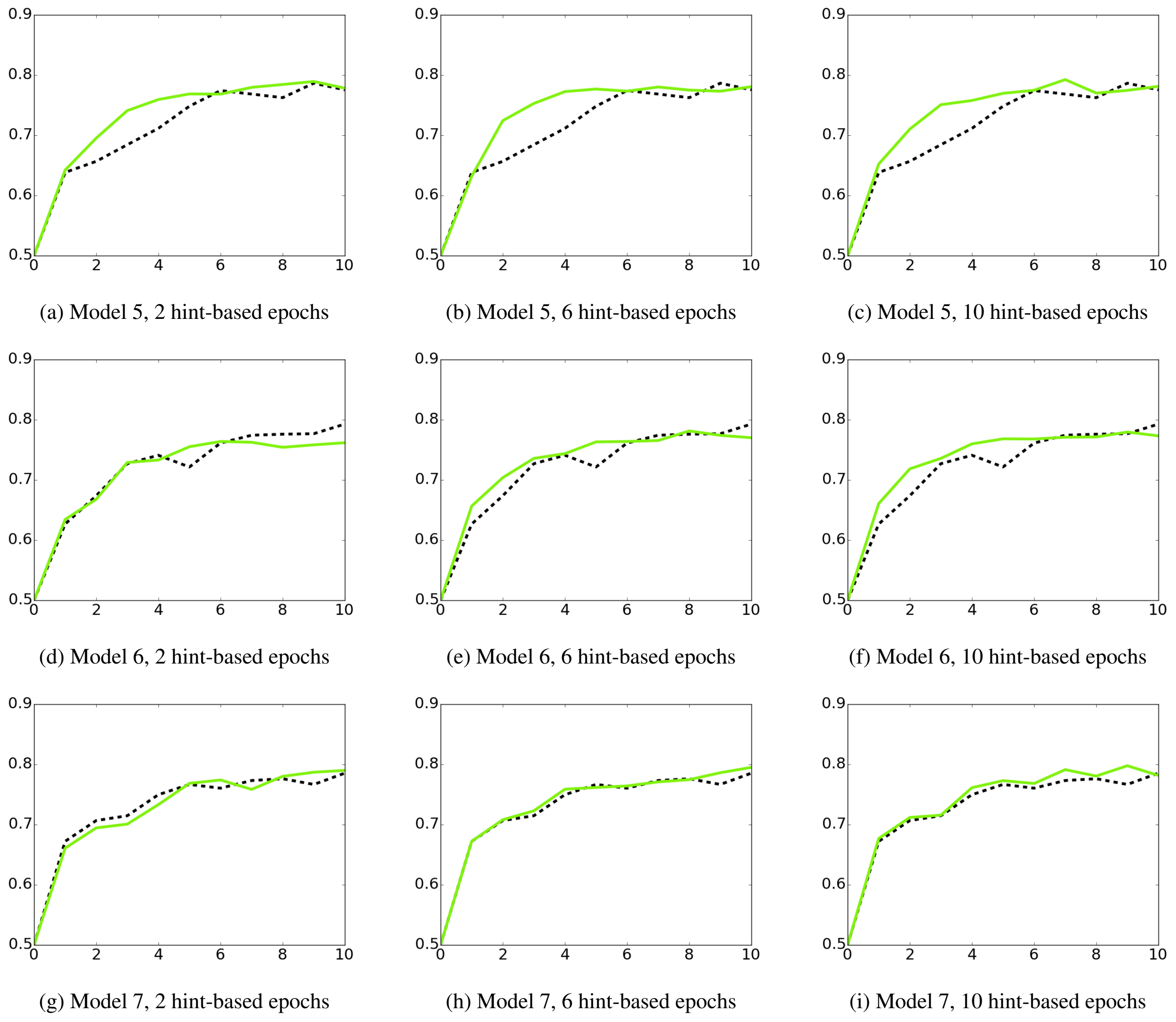
5.2. Validation Accuracy and Time Tradeoffs

We also explored whether our model would improve validation accuracy relative to the amount of time it takes to reach a certain accuracy. The hypothesis is that investing some amount of time in the stage 1 hints-based training to get a good parameter initialization will result in some payoff for the model later on.

Figure 9 shows the validation accuracy for various numbers of epochs used to train model 5 in the stage 1 hint-based training and stage 2 KD-based training. Rows correspond to number of epochs in stage 1 hint-based training, and columns correspond to number of epochs in stage 2 KD-based training. We note that the first row of the diagram is very light. So, no matter how many epochs we spend on the initialization, we definitely need to spend more than a couple epochs training the whole student network before getting any reasonable results.

Model 5 achieved the highest validation accuracy when initialized after training for 5 epochs and when its full network was trained for 7 epochs. We explore the speed of this network's training as well.

Figure 10 plots the validation accuracy against time for which the model has trained. The green line represents Model 5 training for 5 hint-based epochs to initialize its weights, followed by 7 epochs on the full network. The dotted black line shows a control for Model 5 under the vanilla


(a) Model 5, 2 hint-based epochs

(b) Model 5, 6 hint-based epochs

(c) Model 5, 10 hint-based epochs

(d) Model 6, 2 hint-based epochs

(e) Model 6, 6 hint-based epochs

(f) Model 6, 10 hint-based epochs

(g) Model 7, 2 hint-based epochs

(h) Model 7, 6 hint-based epochs

(i) Model 7, 10 hint-based epochs

Figure 8: Accuracy vs. number of epochs the (full) student network has trained. The dotted black line is a control for each model trained without augmented hints.

training framework in OpenFace.

The green line is flat for around 160 seconds. This corresponds to the time it spends in stage 1, training the first half of the network. The green line then sharply grows, steeper than the black dotted line. This shows that augmented hints is paying off once the full network trains. However, the green line never surpasses the black dotted line. So, at no point in training was the increased rate of training worth the time spent in initialization paid up front.

Figure 11 further reinforces the upfront cost of hints-based training. If we consider the ratio of the validation accuracy to the training time as a heuristic for whether the initial investment was "worthwhile," we observe that taking the least amount of time training the guided layer of the student results in the maximum expected payoff. This shows that while this augmented hints may have improved validation accuracy, at no point was it worth the extra time it took.

6. Conclusion

Though the student-teacher networks seem to offset the cost of training for larger network architectures for the student, the approach actually hampered completion of our task. To be usable in a mobile setting, the student network must be small enough to train without the assistance of a hint. For networks in which this is true, guiding a hidden

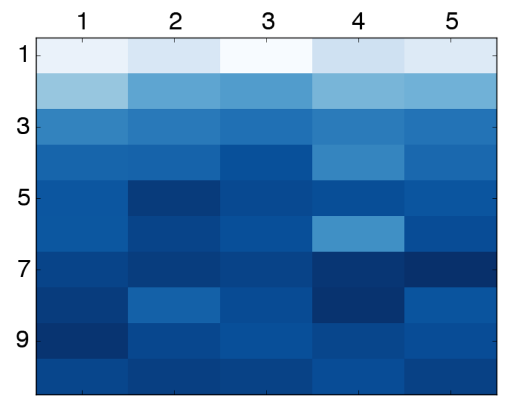


Figure 9: Validation accuracy (darker is higher). Rows correspond to number of epochs in stage 1 hint-based training, columns correspond to number of epochs in stage 2 KD-based training.

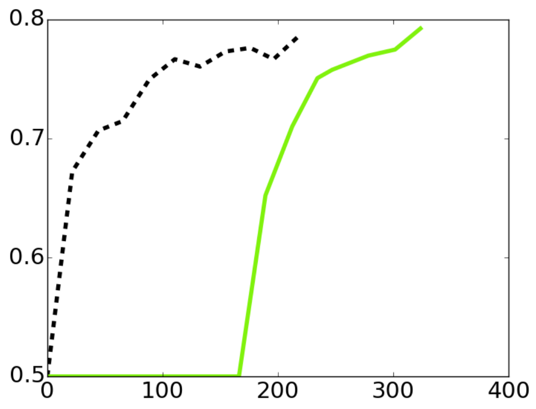


Figure 10: Validation accuracy vs. time (seconds)

layer of the student with a hidden layer of the teacher actually impedes the training process. This is not particularly surprising given the original use case for knowledge distillation proposed by Dean and Hinton–to consolidate knowledge from an ensemble of large networks into a single one.

Our results indicate that the time required to train the student with the teacher does not payoff in terms of validation accuracy for such a small network. In fact, our time would be better served by training the network with vanilla forward and backpropagation.

It follows that deep learning for mobile devices might require student-teacher networks for distillation of information from multiple source networks; however, in the baseline use case of compressing a single network for identity recognition, our results indicate that these methods are somewhat unnecessary.

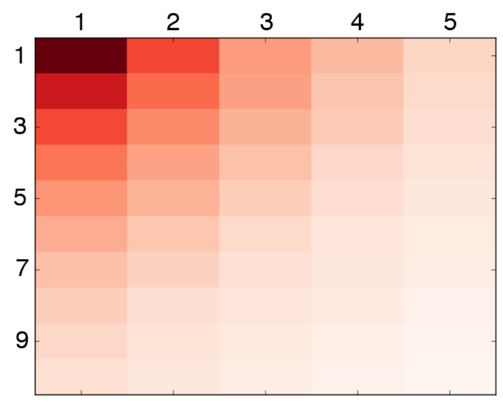


Figure 11: Ratio of validation accuracy to the training time (darker is higher payoff). Rows correspond to number of epochs in stage 1 hint-based training, columns correspond to number of epochs in stage 2 KD-based training.

6.1. Future Work

Future work should focus primarily on improving the speed of the initialization hint-based training phase. However, another question of interest is whether the student architecture can be inherently improved. Suppose we fix a teacher model and provide constraints on the student network, such as the number of parameters the student network can use, the maximum depth of the student network, or the maximum intermediate representation size of the student. Is there a systematic way to find the student network(s) that train the fastest and achieve the same accuracy?

We are also curious about having the student network synchronize with the teacher. The intended use case is for the hints training to occur once to initialize a student network, which can then retrain quickly when new training data comes in. Would it be useful to periodically train the teacher on the most recent batch of new images?

Once we have successfully designed a student-teacher framework that performs well not only as a time of number of initialization epochs required but also as a function of time, we would be excited to test this model on a mobile application for content-based image retrieval, as suggested in the motivation for the project.

Acknowledgements

We thank the OpenFace project team for their open source implementation that allows us to try variants of the FaceNet model in practice. We thank the Labeled Faces in the Wild team for their large and quality dataset.

We thank Nishith Khandwala for allowing us to run our experiments on his computer cluster.

References

[1] B. Amos, B. Ludwiczuk, J. Harkes, P. Pillai, K. Elgazzar, and M. Satyanarayanan. OpenFace: Face Recognition with Deep Neural Networks. `http://github.com/cmusatyalab/openface`. Accessed: 2016-01-11.

[2] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[3] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[4] G. B. Huang, M. Mattar, H. Lee, and E. Learned-Miller. Learning to align from scratch. In *NIPS*, 2012.

[5] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

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

[7] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.