Learning Relevant Information through Knowledge Distillation for Small Dataset

Raj Jagtap
rjagtap@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Pranav Patil
patil127@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Mansi Shinde
mshinde@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Rucha Deshpande
deshpa37@purdue.edu
Purdue University
West Lafayette, Indiana, USA

ABSTRACT

Deep learning models are able to learn relevant features in the dataset. However, these Deep Learning [10] architectures require large labeled datasets for training to achieve state-of-the-art accuracy. Techniques like few shot learning [25], and transfer learning [26] have been majorly used in solving the challenging problem of image classification with a small dataset. However, they sometimes are not able to generalize on test set. Also, models trained on small datasets sometimes learn wrong features in the data to classify an image. In order to solve this issue, we propose to implement Knowledge Distillation [14] [11] to train a shallow model to imitate feature identification of the deep teacher model that is trained on a large dataset and then use perceptron to classify a face in a small dataset.

CCS CONCEPTS

• **Computing methodologies → Object identification**.

KEYWORDS

Knowledge Distillation, Deep Learning, Neural Networks, Image Classification

ACM Reference Format:
Raj Jagtap, Pranav Patil, Mansi Shinde, and Rucha Deshpande. 20XX. Learning Relevant Information through Knowledge Distillation for Small Dataset. In *Proceedings of (Conference acronym 'XX).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Image classification is a widely used method to categorize images and identify objects in images. Methods such as CNN, ResNets [12] have state-of-the-art performance for such tasks. However, the problem with training such deep neural networks is the need for large datasets. One shot learning is a promising strategy to use on problems where images of a class are limited. However, these methods can learn some wrong embeddings to classify the images. For example, in the problem of classifying two faces, suppose only one subject is wearing eyeglasses in all images and the other subject is not wearing eyeglasses in any image. In this case the model might learn to predict based on eyeglasses and not using facial features.

It is evident that deep models trained on large datasets have learned to identify facial features with good accuracy [2]. However, these models may not work very well on small datasets. Image augmentation may lead to overfitting of the dataset.

Another widely used method for datasets with small sizes is few-shot learning. The goal of few-shot learning [25] is to develop optimization techniques and models that can recognize patterns in small datasets with high efficiency[9]. A pre-trained model can generalize over new categories of data (that the pre-trained model has not seen during training) using only a few labeled samples per class utilizing the Few-Shot Learning framework. The main idea behind few-shot learning is to develop a predictor function for similarity. It assesses how similar the two samples are to one another. The similarity function gives 1 if two samples are similar. They return 0 if the samples are different. The learned similarity function can be applied to anticipate results for unseen data after training. The similarity function allows us to evaluate the similarity between the query and each sample in the support set. The sample with the highest similarity rating is then used as the prediction.

Can we combine the best of both worlds? Can we train a small model to classify based on facial features? This is the motivation behind our approach.

We plan to employ Knowledge Distillation to train a shallow model to mimic feature identification of the deep teacher model that is trained on a large dataset and then use perceptron to classify a face in a small dataset.

Traditionally knowledge distillation is a self-supervised methodology. The method focuses on training a smaller network by leveraging the knowledge gained by a bigger teacher network. This is essentially done by taking the labels generated by the teacher network into consideration. An image is passed to both teacher and student networks, and the output of the teacher network -the predicted y label- is used by the student network as true y for training. This ensures that the student learns to mimic the teacher network while having substantially lower depth and number of parameters. Although several methods are proposed across domains that use

Conference acronym 'XX, ,
© 20XX Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

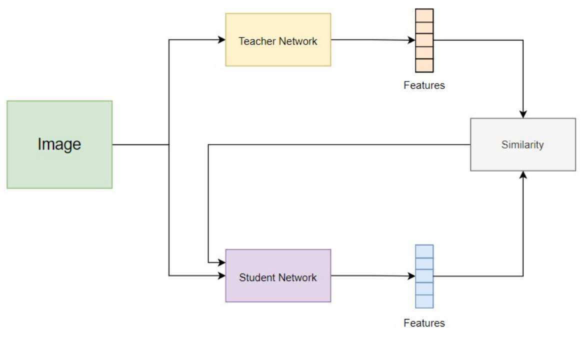

Figure 1: Proposed Model Training Framework

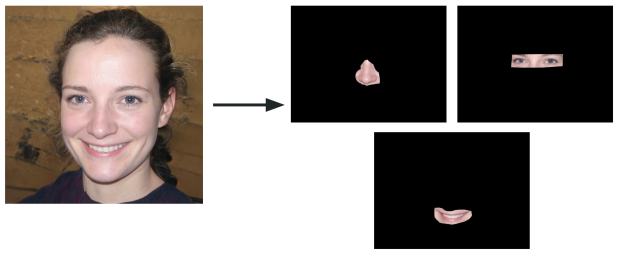

Figure 2: Extracting Feature Images from Face Image

the aforementioned methodology to train the student network, the extent of use of feature-based knowledge distillation still remains comparatively low. Feature-based knowledge distillation is a way of implementing knowledge distillation by making the student learn the features by minimizing the difference between the activations of student and teacher networks.

We plan to employ this version of knowledge distillation for small datasets. We are not interested in the labels predicted by the teacher network, instead, we use the features that the teacher network learns. The student network will be able to match the features that the teacher generates by maximizing the similarity between the features generated by the teacher network and the student network (See Figure 1). In this case, we don't have to take the teacher network that is trained on the dataset into consideration. The teacher network that is trained on identifying if an image is a human face has already learned the features of a human face. We plan on using these features instead of the labels that the teacher network generates.

After this, the shallow student network will be able to predict facial features given any image. Now this shallow student network can be appended with an ANN such that it outputs a label. This combined network can be then trained on original small dataset to alter the weights resulting in the network learning specialized features for the dataset. We will evaluate the if the network is small and learns features can it learn small dataset with good accuracy.

2 RELATED WORK

Deep learning has advanced the field of computer vision significantly, but only under the assumption that they have access to a lot of annotated data. Large-scale data collection is however not feasible in practice. More data is needed for deep learning models to have strong generalization capabilities. Deep learning technology can improve the training effect when there is a lack of data, but the generalization performance of new samples is subpar. Following are some approaches to the problem of image classification on small datasets:

One-shot learning seeks to learn details about various object categories using just a few labeled examples of each type of example. Due to its potential usefulness in practical applications in surveillance security, one-shot learning has attracted the most attention in face recognition and person re-identification tasks. [21] Embeddings for One shot learning are learned using Siamese Network [17] in which a learned feature vector for the known and candidate example are compared. Techniques like Contrastive loss, Triplet loss are used to learn feature embeddings. We implemented One shot learning using Triple loss approach inspired by [22] to learn similarity and dissimilarity among the images.

Finn et al. [8] presented a meta-learning technique based on gradient descent learning of easily adjustable model parameters. The core idea of this method is to train the model's initial parameters to ensure that the model performs as well as possible on a new task once the initial parameters have been changed through one or more gradient steps calculated with a little amount of data from the new task.

Basu et [3] developed Domain Extension Transfer Learning (DETL), a technique that allows for the classification of COVID-19 on chest X-rays using a pre-trained model on NIH Chest X-rays [1]. By reclassifying X-rays into two categories—normal and disease—DETL preprocesses the NIH Chest X-ray dataset. It then pretrains models built using various architectures, such as AlexNet, VGGNet, and ResNet. The performance of the pre-trained models is then compared using a second dataset made up of four classes: normal, other disease, pneumonia, and Covid-19, which was created from a variety of datasets, including the NIH Chest X-rays dataset. With AlexNet, VGGNet, and ResNet, respectively, the models obtain 82.98%, 90.13%, and 85.98% accuracy. This performance is encouraging given that the NIH Chest X-rays dataset has labeling accuracy of >90

Yang et al. [27] modified the Knowledge Distillation formulation and suggest distributed loss to transfer the knowledge of non-target distribution. They suggested soft loss, which regards the teacher's target outcome as a soft goal for the student to achieve. Students can attain cutting-edge performance on the CIFAR-100 and ImageNet by using the proposed New Knowledge Loss (NKD), which comprises distributed and soft loss.

More recently, deep neural networks with concept bottlenecks have emerged as targeted tools for solving particular tasks like content-based image retrieval [4]. Concept Bottle Models are those that bottleneck on human-specified concepts, where the model first predicts the concepts and then uses only those predicted concepts to make a final prediction [18].

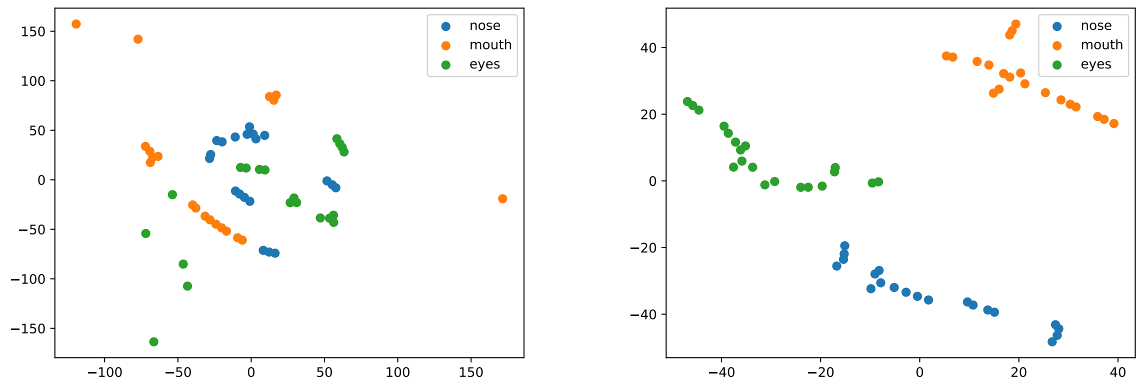

Figure 3:

Scatter Plots of t-SNE on Facial Features Activations.
Left: Activations on Untrained ResNet-50, Right: Activations on ResNet-50 trained on VGGFace2

3 EXPERIMENT

CNNs trained for Facial Features Recognition have shown that deep learning models are capable of learning relevant facial features such as eyes, nose, forehead, mouth, etc. [2]. To test feature learning capability of the ResNet-50 model we perform an experiment. If the model is able to learn different facial features then each feature shall be able to activate a distinct set of neurons in the model.

To perform this experiment, we take images of faces and only select parts of faces namely the nose, eyes, and mouth. Using 20 different images we generate 60 feature images.

We pass 60 images through network and obtain activations vector from the penultimate layer of ResNet-50. ResNet-50 has 2048 neurons in penultimate layer. We convert this vector of 2048 dimensions to 2D using t-SNE [24] and plot a scatter plot. We performed this experiment on two models and compare their scatter plots.

(1) ResNet-50: untrained, randomly initialized
(2) ResNet-50: trained on VGGFace2

Figure 3 shows that a trained model has distinctive activations for different facial features than an untrained model. It is evident from the experiment that a trained model has the capability of learning relevant facial features such as eyes, nose, mouth, etc.

4 DATASET

We use the Pins Face Recognition dataset [5] in this study. The dataset consists of 17,534 faces of 105 celebrities collected from Pinterest. The size of the dataset is 494MB. The images are cropped to keep the face region only. The images are taken in slightly different poses and different lighting conditions.

For our purpose of image classification with less data, we reduce the size of dataset by subsampling only 32 images per class. The total size of our training set is 3360. The images used for classification are of size (224,224). We apply image transformations like rotation, scaling, flipping, gaussian blur, color jitter and random erasing. The transformations are applied by random permutations and combinations with certain probability.

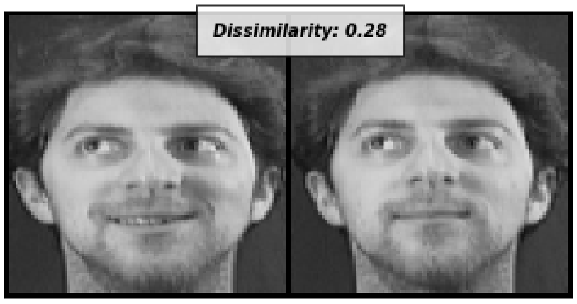

Figure 4: Facial Similarity
1

5 METHODOLOGY

5.1 One Shot Learning

One Shot Learning is an ML-based object classification technique that accesses the similarities and differences between two images. It's to develop the ability to evaluate two images and determine whether or not they are identical. With the use of the fewest possible pictures, it hopes to teach the model how to form its own conclusions about their similarities. Using these instances, a model is created that can forecast the appearance of other unidentified visuals.

One Shot Learning uses the unique class of convolutional neural networks (CNNs) known as Siamese neural networks to function as a similarity comparer (SNNs). Traditional CNNs modify their parameters as they learn to accurately classify each image. Siamese neural networks [17] are trained to evaluate the degree of similarity between features in two input images. This network first takes an input image of a person and determines the encodings of that image. It then uses the same network without making any adjustments to the weights or biases—to predict the encodings of an input image

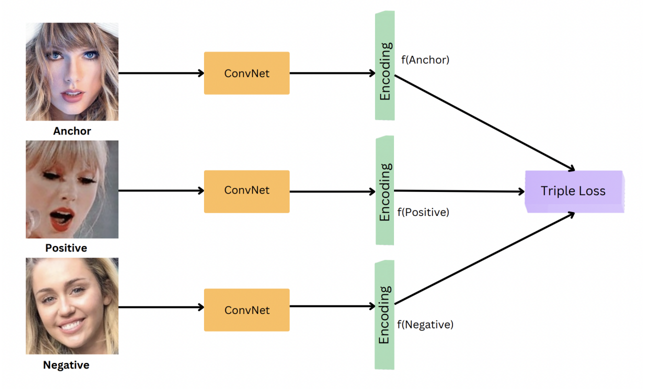

Figure 5: Siamese Network with Triple Loss

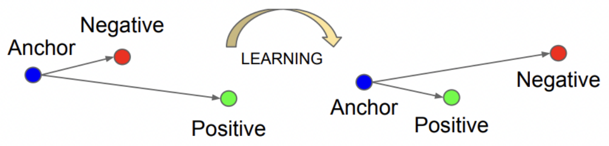

Figure 6: Triple Loss [22]

of a different person. Now it compares these two encodings to see if the two images are comparable to one another. These two encodings function as images' latent feature representations.Images with the same person have similar features/encodings. Using this, we compare and tell if the two images have the same person or not.

We have trained Siamese Neural Network using triple loss approach as done in [22].

The goal of the triplet loss is to make sure that two examples with the same label have their embeddings close together in the embedding space and two examples with different labels have their embeddings far away. To achieve this, we take 3 images as an input to the neural network, anchor image, positive image (image from the same class as the anchor) and negative image (picture from different class to the anchor). With this three triplets, we calculate the loss such that distance between the embeddings of anchor and positive image should be lesser than distance between the embeddings of anchor and negative image.
Triple loss function is deifned as :

$$\mathcal{L}(A,P,N) = max(||f(A)-f(P)||_2^2 - ||f(A)-f(N)||_2^2 + margin, 0) \quad (1)$$

Here, the intuition is that we set the margin (which is a hyper-parameter) such that it pulls $||f(A) - f(P)||_2^2$ closer and pushes $||f(A) - f(N)||_2^2$ farther.

Siamese networks are more robust to class imbalance and they do not need to be retrained to detect new classes, because it evaluates the similarity among the entities per class instead of classifying them. It outperforms in speed accuracy in recognising images, faces or other objects with strong similarity. Though they give such promising results, they possess certain drawbacks which are, require more computation power than other CNNs since there are thrice as many operations needed to teach three models (each for anchor, positive, negative image embeddings) during training, hence more memory consumption. Moreover, SNNs trained for one task can be used for some other task. Also, its accuracy degrades if person in one of the images is wearing hat, scarf or glasses, and the person in the other image is not.

5.2 Model-Agnostic-Meta Learning (MAML)

Meta-learning is a framework where we learn how to classify from training data and evaluate the results using test data. In the traditional learning paradigm, we learn how to classify from training data and evaluate the results using test data. In the meta-learning framework, we learn how to learn to classify given a set of training tasks and evaluate using a set of test tasks. Few-shot learning is generally classified as a meta-learning problem. It's limitation is that it usually generates classifiers that do not generalize effectively because of a lack of data.

Model-Agnostic-Meta Learning is a model and task-agnostic algorithm for meta-learning that trains a model's parameters using a number of gradient updates to achieve quick learning on an unseen task. The aim is to learn a general model that can easily be fine-tuned for many different tasks, even when the training data is scarce. It is an optimization-based meta-learning algorithm which means, it tries to adjust the standard optimization procedure to a few-shot setting. Given a model, support, and query set during training, optimize the model is optimized for m steps on the support set and the gradients of the query loss are evaluated with respect to the original model's parameters. This is done using a few different support-query sets for the same model, and the gradients are accumulated. This results in learning a model that provides a good initialization for being quickly adapted to the training tasks.

Consider a model represented by a parametrized function f_θ with parameters θ. When adapting to a new task T_i, the model's parameters θ become θ_i'. With MAML, the updated parameter vector θ_i' is computed using one or more gradient descent updates on task T_i.

When using one gradient update,

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

The step size α may be fixed as a hyperparameter or metalearned. The model parameters are trained by optimizing for the performance of $f_{\theta_i'}$

with respect to θ across tasks sampled from $p(\mathcal{T}_i)$. The meta objective is as follows:

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right)$$

The meta-optimization is performed over the model parameters θ, whereas the objective is computed using the updated model parameters θ'. Effectively, the purpose of MAML is to optimize the model parameters so that a small number of gradient steps on a new task will result optimal results on the task. Stochastic gradient descent (SGD) is used to do the meta-optimization across tasks so that the model parameters θ are updated as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right)$$

where β is the step size.

5.3 Deep Learning from Scratch

Deep learning has had exponential growth and many architectures have been proposed to get better metrics for the same/similar tasks. We build a convolutional neural network (ConvNet-9) and train on the dataset with the split as mentioned in 4. We also train different state-of-the-art deep neural networks that gave better performance than the benchmark when they were proposed. To explore the models' performance without any prior knowledge (training), we train the following models:

5.3.1 ***ConvNet-9***. (7)

- Convolutional Layers: 9
- Max Pool: 3
- Number of Trainable Parameters: 18M

5.3.2 ***ResNet-50***. [13]

- Convolutional Layers: 48
- Max Pool: 1
- Average Pool: 1
- Number of Trainable Parameters: 23M

5.3.3 ***ResNet-18***. [13]

- Convolutional Layers: 16
- Max Pool: 1
- Average Pool: 1
- Number of Trainable Parameters: 11M

5.3.4 ***Inception V3***. [23]

- Layers: 42
- Max Pool: 4
- Average Pool: 11
- Number of Trainable Parameters: <25M

5.3.5 ***MobileNet***. [15]

- Convolutional Layers: 27
- Average Pool: 1
- Number of Trainable Parameters: 13M

5.4 Deep Learning using Transfer Learning

Training the aforementioned deep models require a lot of resources and with their huge size, they require a proportionally large amount of data. This is not possible for small datasets given a limited amount of resources. Thus, we use transfer learning by training the pre-trained models on the dataset of our interest. We take the output of convolutional layers of the pre-trained models and add our classifier layers. We freeze the convolutional layers to retain their already trained parameters and only fine-tune the weights of the final layers of the model during training on the dataset of our interest. This helps us to leverage the features learned by the models from the dataset they were originally trained on. We train following models using transfer learning:

5.4.1 ***ResNet-50:*** Pre-trained on: VGG-FACE [20]

5.4.2 ***Inception V3:*** Pre-trained on: ImageNet[7]

5.4.3 ***MobileNet:*** Pre-trained on ImageNet[7]

The deep learning methods show promising results but still training massive networks raises concerns about computational resources. While the dataset in consideration is small and these deep models can be trained rather quickly, this is not always the case. Also, these models are prone to memorizing the data if the amount of data is less, leading to overfitting. With the rise of big data and the increase in the number of edge devices using AI to predict, the size of networks and consequently computations required for training and prediction play a huge role in deployment.

5.5 Knowledge Distillation with Labels

Knowledge distillation[14] is a method of model compression by training a student network using the actual dataset and the predictions given by a pre-trained teacher network. The goal is the make the student model mimic the teacher network's predictions. Since the teacher is trained on the same dataset and thus has learned the relevant features, we can use these features and labels and *"distill"* the knowledge that the teacher network has learned to the student network.

In this method, we use the predictions from the teacher network directly to train the student network. The goal is to predict the teacher's final predictions directly. We train the teacher network on the dataset in consideration using transfer learning (i.e the models trained in the previous method). We pass the image through the teacher network and get the predictions for every class. Then these predictions are used in addition to a temperature parameter T (set to 1 in our case) to calculate the loss of the prediction the model gives for the same image. We use categorical cross-entropy as a loss function for distillation.

$$\mathcal{L}_{cc} = -\sum_{i}^{C} \hat{t}_i log\left(\frac{e^{\hat{s}_i}}{\sum_j^C e^{\hat{s}_j}}\right) \tag{2}$$

where C is the number of classes, $\hat{t}_i$ is the probability prediction by the teacher network corresponding to the class i, and $\hat{s}_i$ is the probability predicted by the student network corresponding to the class i. To retain the actual labels, we also use a hyperparameter α that controls the weightage given to the loss corresponding to the actual labels and the labels from the teacher network.

$$\mathcal{L} = \alpha\mathcal{L}'_{cc} + (1-\alpha)\mathcal{L}_{cc} \tag{3}$$

where $\mathcal{L}'_{cc}$ is the categorical cross-entropy loss against the actual labels.

$$\mathcal{L}'_{cc} = -\sum_{i}^{C} y_i log\left(\frac{e^{\hat{s}_i}}{\sum_j^C e^{\hat{s}_j}}\right) \tag{4}$$

y_i being one or zero based on the image being of the class i or not.

We use this method with student network as ConvNet-9 and use Inception-V3[23] (trained on Imagenet[7] and Pins Dataset [5]) and Resnet-50 [13]

5.6 Knowledge Distillation with Features

In this method, we train the student network to learn features of the penultimate layer learned by the teacher network. Since we showed in the section 3 Experiment that a ResNet-50 model trained on a large dataset such as VGGFace2 is capable of learning facial

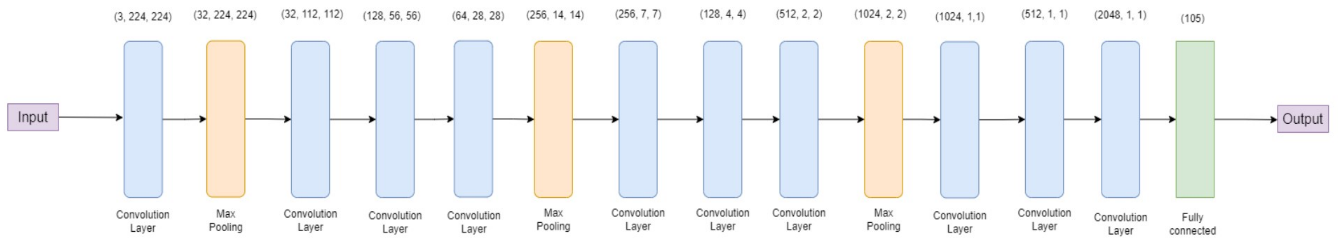

Figure 7: ConvNet-9 Architecture

features, this method is an attempt to distill such relevant features and train the student network.

In this method, we cut the networks before the last classifier layer. The training pipeline of this method can be broken down into two following parts

(1) **Pre-training:** Train first part of student model using activation generated by first part of teacher model.
(2) **Fine-tuning:** Freeze learnable parameters of first part os student network and finetune second part (last layer) using the labeled dataset.

Since in the pre-training step, we are training student network only to learn activations of penultimate layers of teacher network, we do not require any labeled dataset. For this step, we create images of faces using StyleGAN [16] with random seeds and train with these images as input to networks. We obtain activations of ResNet-50 and use them as targets for training the student network.

In the fine-tuning step, we freeze weights of part 1 of student network and train the last layer using the labeled dataset. The weights are frozen to avoid any updates to the part of the network that mimics the activations of the teacher model. We also experimented without freezing weights, this leads to adverse updates to the first part of the student networks and the performance is not much better than training from scratch.

In this section, we describe in detail the two methods we use in the pre-training step to train different student networks using ResNet-50 as teacher network.

5.6.1 ***Training as Regression.*** Since we want to train student models to mimic activations of teacher network, we consider this as a regression problem and proceed to train using Mean Squared Error (MSE) as loss function.

We obtain activations from teacher network and activations from student network and calculate MSE for each batch as follows:

$$\mathcal{L} = \frac{1}{b} \sum_{i=0}^{b} \|\hat{y}^{(i)} - \overline{y}^{(i)}\|_2^2$$

where $\hat{y}$ is the activation vector from student network, $\overline{y}$ is activation vector from teacher network and b is the batch size. Then update the weights of student network to minimize $\mathcal{L}$.

Using this training method we train ConvNet-9 and ResNet-18 to mimic activations generated by the ResNet-50 model pre-trained on the VGGFace2 dataset.

5.6.2 ***Training as Multi-Label Classification.*** This method is inspired from Concept Bottleneck Models [19].

In this method, we make a modeling assumption that instead of learning for feature values of teacher network, we train for probabilities of concepts to be present in the given image. For this method, we modify the teacher network by adding a connector linear layer with Sigmoid activation between the penultimate layer and the last layer. This way we ensure that all the outputs of the new penultimate layer are between 0 and 1. This, however, modifies the original teacher network that was trained on VGGFace2 dataset. Also, since a new randomly initialized layer is added in the network we fine-tune the new layer of teacher ResNet-50 network using the labeled dataset in similar way as Transfer Learning. We do not see a significant drop in the performance of the teacher model as compared to transfer learning. Hence we proceed with Knowledge Distillation of Activations considering problem as a Multi-Label Classification.

We obtain activations from teacher network and activations from student network and calculate Binary Cross Entropy Loss for each batch as follows:

$$\mathcal{L} = -\frac{1}{b} \sum_{i=0}^{b} \left[\overline{y}^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - \overline{y}^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right]$$

where $\hat{y}$ is the activation vector from student network, $\overline{y}$ is activation vector from teacher network and b is the batch size. Then update the weights of student network to minimize $\mathcal{L}$.

Using this training method we train ConvNet-9 and ResNet-18 to mimic activations generated by the ResNet-50 model pre-trained on the VGGFace2 dataset.

6 EXPERIMENTAL SETUP

6.1 One Shot Learning

We implemented One Shot Learning with triple loss calculation in Pytorch[2]. We can use any pretrained or general Convolutional Neural Network as a Siamese Network. We used Convolutional Neural Network with 9 layers and Pretrained Resnet-50 model as a Siamese Network.

First, we generate the embeddings of the images and create a triplet set of images where in each triplet, Anchor and Positive image belongs to same class and Negative image belongs to different

[2]https://pytorch.org/

class. Further, we feed this triplet to Siamese Network. Inorder to train Siamese Network, we use Euclidean Distance to calculate the distance between anchor and positive image embeddings and, anchor and negative image enbeddings. Then, calculate the triple loss using below equation:

$$\mathcal{L}(A,P,N) = max(||f(A) - f(P)||_2^2 - ||f(A) - f(N)||_2^2 + margin, 0) \tag{5}$$

Finally compute gradient using Adam Optimizer and backpropagate to update the weights.

Accuracy in one-shot triple loss is calculated as proportion of the number of triplet in which feature distance between anchor and positive is less than that between anchor and negative.

$$A(A,P,N) = \sum_i (||f(A_i) - f(P_i)||_2^2 < ||f(A_i) - f(N_i)||_2^2) \tag{6}$$

where i is the number of triplets.

We ran our one-shot model for 100 epochs with Convolutional Neural Network as a Siamese Network and for 50 epochs with Pretrained Resnet-50 as a Siamese Network, and further calculated the loss and accuracy.

6.2 Deep Learning Models

We implement deep learning models and training pipeline in PyTorch [3]. The ResNet-50 model code is used from GitHub of Cydonia999 [4]. Pre-trained ResNet-50 weights are converted from weights provided by VGGFace2 authors [5]. Our implementation includes a common training pipeline such that the only difference is the model to be trained and a hyper-parameter learning rate. This helps avoid any performance gain or degradation based on training method. For example, a model trained with regularization and another model trained without regularization may have different performances, but since regularization is an additional factor in training, it is hard to infer if the performance is due to the model or the regularization.

In this experiment, we fix following parameters

(1) Train-test split images
(2) Batch size = 16
(3) Image pre-processor
(4) Loss function = Binary Cross Entropy
(5) Scoring functions
  (a) F1 Score (micro & macro averaging)
  (b) Precision (micro & macro averaging)
  (c) Recall (micro & macro averaging)
  (d) Accuracy
(6) Step size function

We use diminishing step size as follows:

$$lr_t = \frac{lr}{\sqrt{1+t}}$$

where lr is initial learning rate and lr_t is learning rate at epoch number t.

Using the above setup we train models from scratch and using transfer learing for 200 epochs. We tune the hyper-parameter initial learning rate lr because an optimal learning rate for one model may overshoot updates for another model and may have very slow convergence for another model.

[3]https://pytorch.org/
[4]https://github.com/cydonia999/VGGFace2-pytorch/blob/master/models/resnet.py
[5]https://github.com/cydonia999/VGGFace2-pytorch#pretrained-models

6.3 Knowledge Distillation with Labels

(1) Teacher Network: Inception V3
(2) Student Network: ConvNet-9
(3) Batch Size: 32
(4) Preprocessor: Inception Preprocessor

6.4 Knowledge Distillation with Features

We implement the Knowledge Distillation training pipeline in PyTorch. Our implementation includes a two-step learning pipeline that trains the first part of student model using Knowledge Distillation, freezes the learnable parameters of first part, and fine-tunes the second part using the labeled dataset. This pipeline is also designed to avoid any additional factors that may affect performance of training.

For the pre-training step, we generate 102, 400 images with random seeds 0 to 102399 using StyleGAN and save them. We use these images as dataset for training teacher network activations to student networks.

For the pre-training step we fix following parameters

(1) batch size = 32
(2) Image pre-processor
(3) Loss functions
  - Mean Squared Error for Regression (Refer Sec. 5.6.1)
  - Binary Cross Entropy Loss (Refer Sec. 5.6.2)
(4) Scoring functions
  (a) Mean Squared Error
  (b) Mean Absolute Error
  (c) R2-score
(5) Step size function

For the fine-tuning step we fix following parameters

(1) Train-test split images
(2) Batch size = 16
(3) Image pre-processor
(4) Loss function = Binary Cross Entropy
(5) Scoring functions
  (a) F1 Score (micro & macro averaging)
  (b) Precision (micro & macro averaging)
  (c) Recall (micro & macro averaging)
  (d) Accuracy
(6) Step size function

For both of the steps, we use diminishing step size as follows:

$$lr_t = \frac{lr}{\sqrt{1+t}}$$

where lr is initial learning rate and lr_t is learning rate at epoch number t.

7 RESULTS AND ANALYSIS

7.1 One Shot Learning

We build the Siamese Network using below 2 approaches:

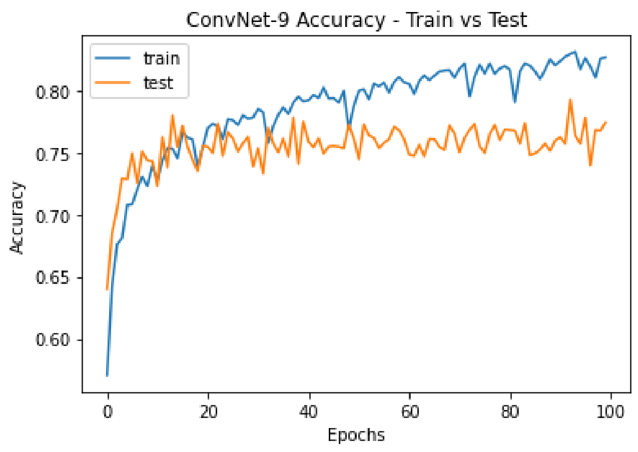

Figure 8: OneShot ConvNet-9 Accuracy

(1) Base ConvNet-9 with 9 layers and initialized with random weights
(2) Pretrained Resnet-50 model initialized with weights trained on the VGGFace2 dataset

And then further trained them using Triple Loss.

Model	ACC
ConvNet-9 lr=0.001 epoch=100	83.10
Resnet-50 lr=0.001 epoch=50	91.16

Table 1: Accuracy of One Shot using CNN and Resnet-50 models

From the Table 2 we can observe that:

- Highest accuracy is achieved by using ResNet-50 model as Siamese Network as compared to ConvNet-9
- this is because ConvNet-9 has only 9 layers hence model might not be able to learn relevant features.

7.2 Deep Learning from Scratch

Since we are training from scratch, all the models are initialized with random weights.

From the Table 2 we can observe that

- Highest F1 score is achieved by ResNet-18.
- ResNet-50 performs well on training set but does not perform good on test set.
- This is because ResNet-50 has $\approx 23.5M$ learnable parameters that are trained on dataset of size 3360. This leads to heavy overfitting.
- ResNet-18 also has $\approx 11.4M$ learnable parameters hence model overfits but overfitting is not as prominent as ResNet-50.

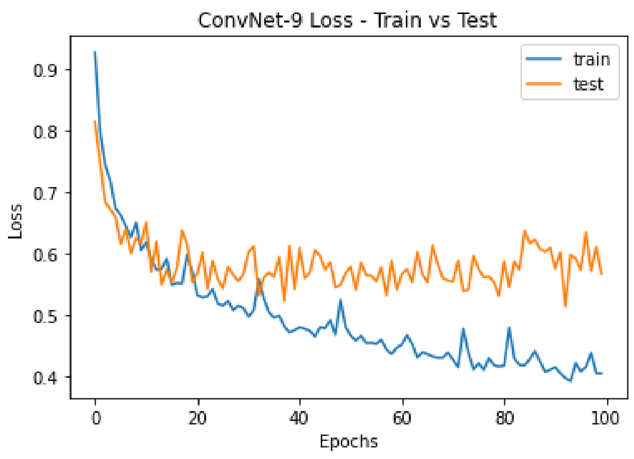

Figure 9: OneShot ConvNet-9 Loss

Model	F1-score	PRE	REC	ACC
ResNet-50 lr=0.001 epoch=100	95.17 40.34	95.24 42.75	95.18 41.37	95.18 40.09
ResNet-18 lr=0.001 epoch=185	99.85 52.38	99.85 53.33	99.85 53.58	99.85 52.14
ConvNet-9 lr=0.001 epoch=200	97.26 20.25	97.34 20.30	97.26 21.05	97.26 19.78
InceptionV3 lr=0.01 epoch=272	99.89 70.49	99.89 74.91	99.89 66.67	99.3 68.77
MobileNet lr=0.01 epoch=71	100 69.75	100 74.41	100 65.73	100 68.15

Table 2: Performance of Deep Learning Models Trained from Scratch. Epoch of Top-1 F1-Score is presented. PRE, REC, and ACC denote Precision, Recall, and Accuracy, respectively (macro averaging).

- Simple ConvNet-9 also performs well on training set but does not perform on test set. This can be because
  (1) ConvNet-9 has only 9 layers, hence model may not be able to learn relevant features.
  (2) Overfitting due to $\approx 18M$ parameters.

7.3 Deep Learning using Transfer Learning

In transfer learning, we initialize model weights with pre-trained model weights. In our experiment ResNet-50 is initialized with weights trained on the VGGFace2 dataset and Inception-V3 is initialized with weights trained on the ImageNet dataset. It is important to note that ResNet-50 is trained on a dataset that is closer to the dataset we use than the dataset used by InceptionV3. Also, it is important to note that ResNet-50 achieves a score of 90.8% with 0.001 false positive rate [6]

Model	F1-score	PRE	REC	ACC
ResNet-50 lr=0.001 epoch=58	99.40 97.71	99.42 97.70	99.40 97.79	99.40 97.71
InceptionV3 lr=0.01 epoch=242	100 26.12	100 31.48	100 22.45	100 26.09
MobileNet lr=0.01 epoch=56	100 37.16	100 46.72	100 31.02	100 36.57

Table 3: Performance of Deep Learning Models Trained using Transfer Learning. An Epoch of Top-1 F1-Score is presented. PRE, REC, and ACC denote Precision, Recall, and Accuracy, respectively (macro averaging).

From the Table 3 we can observe that

- Highest F1 score is achieved by ResNet-50. This is partly because, ResNet-50 was the only model that was pre-trained on a dataset of faces.
- ResNet-50 also performs best on test set, which implies that the pre-training weights are able to generalize well.

7.4 Knowledge Distillation with Labels

We train ConvNet-9 using knowledge distillation using the Inception-V3 as teacher network. We set hyperparameter $alpha = 0.1$ and temperature parameter $T = 1$.

Model	ACC
ConvNet-9 lr=0.001 epoch=89	81.41 2.561

Table 4: Performance of knowledge distillation using Inception-V3 as teacher network and ConvNet-9 as student network

We find that the performance of the model is quite less compared to it trained independently from scratch. The model is clearly overfitting and thus, training the model with just the label values is not as effective as training the model only on the training dataset.

7.5 Knowledge Distillations with Features

Since ResNet-50 performs best on test set (Refer 7.3), we choose ResNet-50 as the teacher network for Knowledge Distillation of Features.

We initialize student models with random weights and train in two steps as described in Section 5.6 and Section 6.4.

Model	F1-score	PRE	REC	ACC
ConvNet-9 lr=0.001 epoch=86	0.92 0.79	1.04 0.88	2.58 2.39	2.58 2.29

Table 5: Performance of Deep Learning Models Trained using Knowledge Distillation with MSE Loss. Epoch of Top-1 F1-Score is presented. PRE, REC, and ACC denote Precision, Recall, and Accuracy, respectively (macro averaging).

7.5.1 *Training as Regression*.

Here, we pre-train part 1 of student network with MSE as loss function.

Based on the Table 5 we can observe that the F1-score for ConvNet-9 model is very less. The accuracy is only 2.29 on test set. Since the dataset has 105 classes, a dummy classifier that predicts one class all the time will also have an average accuracy of $\frac{1}{105} = 0.95\%$. This means the model is not able to learn any relevant features from the teacher model.

To find the problem with model we check the similarity of activations of student and teacher networks using R2-Score metric. An R2 score can take any value between $-\infty$ to 1, where 1 means model is able to mimic targets perfectly. R2-score of 0 can be achieved by a dummy model that only outputs same vector irrespective of any input. An R2-score can also be negative because a model can be arbitrarily bad at predicting target vector. Our model was able to achieve an R2-score of -4062 at 100'th epoch of pre-training step. This implies that the models is not able to learn from teacher network.

Similarly, ResNet-18 achieves highest R2-score of -20.61 at epoch number 82. Also reducing learning rate to 0.0001 and 0.00001 do not have a significant effect on R2-score. Best R2-score achieved by ResNet-18 was -14.41.

This suggests that performing Knowledge Distillation of activations using MSE is not effective. This can be because ResNet-50 has 2048 activations in penultimate layer and all the values taken by them are in between 0 and 15. Since in MSE, we always consider mean of distances in every dimension the model may show small loss when all values of predicted vector is average of the target vector.

To overcome this we want a loss function that can give importance on each element of the predicted vector to be similar to the corresponding element in the target vector.

7.5.2 *Training as Multi-Label Classification*.

To overcome above-mentioned problem, Binary Cross Entropy seems to be a promising loss function as it focuses on matching probability distribution of predicted and target vectors. However, BCE only works with probability distributions. Hence, we convert we make a modeling assumption that the activations of teacher network are probabilities of some attributes to be present or not in the image.

Hence we make modifications in teacher and student network as mentioned in Section 5.6.2 and train the student networks with BCE as loss function.

From the Table 6 we can observe that

- ResNet-18 achieves F1-score of 79.83% on test set.

Model	F1-score	PRE	REC	ACC
ResNet-18 lr=0.001 epoch=200	81.96 79.83	82.49 80.03	81.66 80.62	81.93 79.83
ConvNet-9 lr=0.001 epoch=200	52.72 45.24	52.99 46.92	53.06 46.50	53.06 44.80

Table 6: Performance of Deep Learning Models Trained using Knowledge Distillation with BCE Loss. Epoch of Top-1 F1-Score is presented. PRE, REC, and ACC denote Precision, Recall, and Accuracy, respectively (macro averaging).

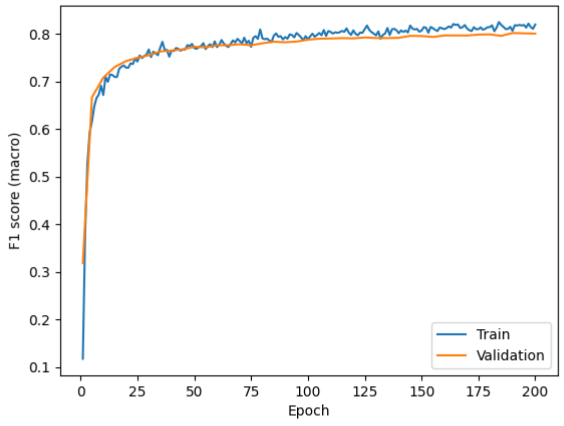

Figure 10: ResNet-18 F1-score vs Epoch

- This implies that ResNet-18 is able to learn relevant facial features as learned by teaher network ResNet-50.
- ConvNet-9 also performs well with an F1-score of 45.24% on test set.

We were able to improve the performance of models as compared to training form scratch. This was possible without need of any additional labeled dataset for training. We were able to pre-train models using only input data that matches input data of the original dataset.

This shows that smaller models were able to learn relevant features upto a certain presicion using Knowledge Distillation.

8 CONCLUSION

We explored different methods of classification for small datasets. We experimented with One-Shot Learning and Deep Learning models. Deep Learning models require a large dataset to generalize learnings. However, on a small dataset we observed training deep learning models from scratch leads to overfitting and bad performance on test set.

To overcome the problem of generalization with Transfer Learning using models pre-trained on large datasets. However, deep learning models are heavy for deployment on edge devices due to their size and number of computations performed per prediction.

Hence, we attempted to reduce the size of models by training small models to mimic relevant feature learnings of deep models using Knowledge Distillation. In this experiment, we were able to train a model (ResNet-18) that is approximately half the size of deep model with a reasonable performance on test data.

Limitations and Future Work

The knowledge distillation method used in this paper uses an unlabeled dataset for pre-training. However, this type of dataset is not always possible to obtain. Future work can include a generative method to create a large unlabeled dataset that is similar to small dataset.

REFERENCES

[1] 2017. NIH Clinical Center provides one of the largest publicly available chest X-ray datasets to scientific community. https://www.nih.gov/news-events/news-releases/nih-clinical-center-provides-one-largest-publicly-available-chest-x-ray-datasets-scientific-community

[2] Shima Alizadeh and Azar Fazel. 2017. Convolutional Neural Networks for Facial Expression Recognition. (04 2017).

[3] Sanhita Basu, Sushmita Mitra, and Nilanjan Saha. 2020. Deep learning for screening covid-19 using chest x-ray images. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2521–2527.

[4] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. 2018. Semantic bottleneck for computer vision tasks. In *Asian Conference on Computer Vision*. Springer, 695–712.

[5] Burak. 2020. Pins face recognition. https://www.kaggle.com/datasets/hereisburak/pins-face-recognition

[6] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. 2018. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*. IEEE, 67–74.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[9] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez Pérez, and Matthieu Cord. 2019. Boosting Few-Shot Visual Learning With Self-Supervision. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 8058–8067. https://doi.org/10.1109/ICCV.2019.00815

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[11] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. 2020. Knowledge Distillation: A Survey. *CoRR* abs/2006.05525 (2020). arXiv:2006.05525 https://arxiv.org/abs/2006.05525

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. 770–778. https://doi.org/10.1109/CVPR.2016.90

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. https://doi.org/10.48550/ARXIV.1503.02531

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[16] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8110–8119.

[17] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, Vol. 2. Lille, 0.

[18] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept bottleneck models. In *International Conference on Machine Learning*. PMLR, 5338–5348.

[19] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept Bottleneck Models. *CoRR*

abs/2007.04612 (2020). arXiv:2007.04612 https://arxiv.org/abs/2007.04612

[20] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep Face Recognition. In *British Machine Vision Conference*.

[21] Candida S. Punla, https://orcid.org/ 0000-0002-1094-0018, cspunla@bpsu.edu.ph, Rosemarie C. Farro, https://orcid.org/0000-0002-3571-2716, rcfarro@bpsu.edu.ph, and Bataan Peninsula State University Dinalupihan, Bataan, Philippines. 2022. Are we there yet?: An analysis of the competencies of BEED graduates of BPSU-DC. *International Multidisciplinary Research Journal* 4, 3 (Sept. 2022), 50–59.

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2818–2826. https://doi.org/10.1109/CVPR.2016.308

[24] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[25] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)* 53, 3 (2020), 1–34.

[26] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big data* 3, 1 (2016), 1–40.

[27] Zhendong Yang, Zhe Li, Yuan Gong, Tianke Zhang, Shanshan Lao, Chun Yuan, and Yu Li. 2022. Rethinking knowledge distillation via cross-entropy. *arXiv preprint arXiv:2208.10139* (2022).