Low-Shot Learning for Face Recognition

Ramya Sarma
University of Massachusetts
Amherst, MA
rsarma@umass.edu

Ananya Ganesh
University of Massachusetts
Amherst, MA
aganesh@umass.edu

Abstract

Low shot learning, or the ability to learn from a small number of examples, is a relevant problem in the domain of facial recognition, where access to training data is limited. We explore a solution based on data augmentation by hallucinating new examples, which was found to work well for classification on ImageNet. We describe the performance of this augmentation strategy for low shot learning on the Microsoft Celeb Faces dataset in comparison to a reasonable baseline, which we beat by 4 percentage points. We also discuss the challenges presented by our data and an analysis of what worked and what didn't.

1. Introduction

Face recognition is the task of uniquely identifying a person, given an image of their face. It is an important problem in security systems, access control, automated authentication, etc. This can be modeled as a classification task where each class is a person whose identity is known. With enough example images for each person, a classifier can be trained using handcrafted features [1] or using learned features [12]. Deep learning approaches to face recognition are found to be very effective, but a significant drawback is that they require large labeled datasets in order to achieve good performance. This is a problem because access to new images of faces might be limited or expensive, along with additional issues like privacy.

To combat this problem, we explore a low-shot learning approach to face recognition. Low-shot learning is the ability to learn to distinguish classes from a small number of examples. We focus on low-shot visual recognition where AI systems are taught to recognize different objects from images using very few examples. In the low-shot learning set-up, there are a fixed number of base classes, for which a large number of training examples are available, and then there are novel classes, for which a limited number of training examples are available. The classifier is then evaluated based on its ability to correctly classify even the novel classes. The key idea is that the rich information provided by the base class data helps to improve the recognition accuracy in cases where we have lesser training data, that is the novel classes.

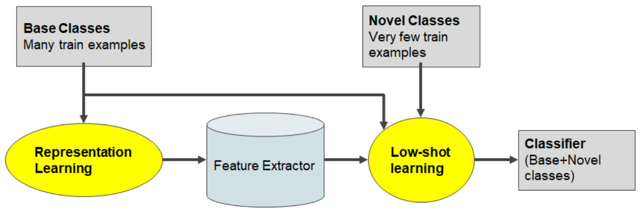


Figure 1. Low shot learning approach

For object recognition, a low-shot learning method based on shrinking and hallucinating features was proposed by Hariharan and Girshick in [6], which achieved good results on the ImageNet classification challenge. We will be extending their approach to the task of face recognition to evaluate its applicability in a drastically different domain. The solution consists of two phases, as described in Figure 1. First, features from the base class are learnt by a feature extractor such as a convolutional neural network. Then, in the low shot learning phase, a classifier is trained on features from both the base and novel classes. The accuracy of this classifier can then be improved by generation strategies, one of which is a novel method of 'hallucinating' training data, proposed in [6]. The intuition behind hallucination is that variations within a class can be considered as transforms, which can then be applied on other classes in order to generate new examples. We describe this in more detail in Section 4.

We use a subset of images of celebrities available on the web as training data, which was compiled by Microsoft Research in 2016 as a benchmark for face recognition [5]. We hope that this technique contributes to various real-world applications, such as image captioning and news video analysis. For this report, we describe our dataset and the preprocessing involved, our technical approach for augmentation, challenges, and future directions.

2. Related Work

As mentioned in our introduction and abstract, our main approach is to implement low-shot learning by hallucinating features, which was proposed by Hariharan and Girshick [6] in 2016 at Facebook AI Research. We also briefly discuss other work in the field of low shot learning as well as face recognition.

2.1. Face recognition

The most impactful work in face recognition, to our best knowledge, was by Turk and Pentland [15] in 1991, which projects face images to a feature space defined by the eigenvectors of the set of faces, called ”eigenfaces”. Their approach is unsupervised, but also requires handcrafted features, which more recent approaches have moved away from. Zhao, et al., in 1998 [18], propose a similar approach based on Linear Discriminant Analysis and Principal Component Analysis which reduce the projected feature space, but also requires handcrafted features.

More recent models such as that by Huang, et al. [8] use a combination of handcrafted features such as local binary patterns (LBP) along with supervision to achieve state-of-the-art results. [3] uses transfer learning to apply a Joint Bayesian model [4] learnt on a large dataset and evaluate on the Labeled Faces in the Wild benchmark [9] (LFW).

DeepFace [12] by Facebook AI Research proposed a deep learning model that requires no handcrafted features and only uses features learnt by a nine layer deep neural network. They achieve close to human level performance on the LFW benchmark.

2.2. One-shot and Low-shot learning

For one shot learning, Siamese networks [10] have been widely adopted for a range of tasks. [10] Approached the one-shot object recognition problem by modeling a Siamese Network architecture for image verification. The Siamese network is trained to verify if a pair of images belong to the same class or not. The generic dataset is used to train the network on the verification task. During inference, the single training sample available for each one-shot class is presented along with the test image and verified if they belong to same class or not.

Bertinetto et al. [2] suggested a second deep network after a network, called a learnet to predict parameters in a previous network. They argued that it is applicable in 1-shot learning, however, the result was not satisfactory. Meanwhile, Wang and Herbert [17] trained a paired network, one that learns few samples with annotated, categorized few dataset, and one with larger samples. With a premise that transform is a type of regression, they learned a transform function with a multilayered regression neural network. We use a similar method to train our generator. Similarly, Vinyals et al. [16] trained a matching network, but with small labeled support set and large unlabeled dataset. These ideas were successful in terms of transferring the knowledge to other dataset.

For face recognition specifically, [11] finds a unified embedding for faces in a space where Euclidean distance between two points correspond to a similarity measure between the two corresponding faces. They use a deep convolutional Neural Network trained on Triplet loss to achieve this.

2.3. Transfer learning

Transfer learning [14], first introduced by Thrun, utilizes knowledge extracted and saved from a specific topic to help the learning on the other similar field. Although a large amount of data is required in at least one domain, the model can use this knowledge to generalize to another domain where less data is available. This is most commonly used by fine-tuning models trained on ImageNet. Transfer learning for faces was attempted by [13] who find that low dimensional features work best.

3. Datasets

We used a subset of the 1 Million Celebrities dataset from Microsoft Research [5]. As described in the introduction, it comprises of face images collected from the internet. The 1M Celeb Faces dataset is designed for two benchmark tasks: a classification challenge across one million categories, which they describe as the largest classification problem in computer vision till date. The second benchmark task focuses on low shot recognition for facial recognition, and provides training data partitioned into base and novel classes. They provide 20,000 base classes (people) with 50-100 training images per person. They also provide 1000 novel classes with 1-5 images per person for training. The test set consists of between 10-20 images per person. Our original intention was to evaluate on this low shot benchmark, but we noticed several problems with the data that they provide.

The complete dataset consists of complex and unprocessed images and the data distribution within this celebrity 1M dataset is not uniform. Also, the images are not aligned or cropped. Hence, the dataset originally provided for the low-shot learning benchmark task warrants us to run face detection along with face recognition algorithms, which we felt was beyond the scope of our project. The dataset that we use is a cleaner subset of the 1 Million Celebrities dataset which has been cropped and aligned. This was originally provided as a sample to look at before

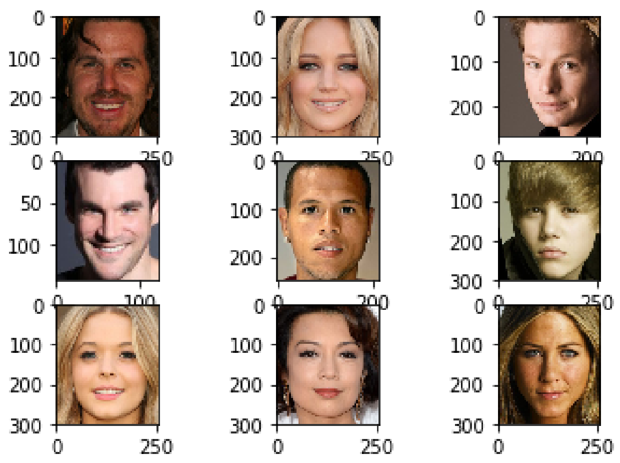


Figure 2. Sample image from each base class

downloading the bigger dataset and we thought this would be a useful starting point to debug our approach. We use this dataset instead of the 1 Million Celebrities dataset to eliminate the challenges presented by scalability and noisy samples and focus on low-shot learning.

We create our own low-shot learning scenario as follows: our base classes consists of 9 different celebrities, and we have 50 training images and 10 test images per class. However, we had to form novel classes by manually collecting images from the internet, which we then cropped and aligned. This resulted in 5 novel classes, with 5 training images and 10 test images per class. Sample images from the base classes are shown in Figure 2. All our input images are scaled to size 300x300 and also normalized.

4. Technical Approach

Our key idea is to improve classification accuracy by generation. In order to assess if generation actually helps with our data, we ran a preliminary experiment based on a naive jittering strategy. Since the results from this experiment were promising, we then implemented the hallucination approach from the Hariharan and Girshick paper. We outline both approaches below.

4.1. Preliminary Approach

To examine the performance of a powerful classifier when the amount of data is limited, we train a convolutional neural network end-to-end on first the base classes, and then the entire training data. This is also used as a baseline for evaluation. For our baseline model, we train the ResNet10 [7] architecture which stands as the state-of-the-art architecture for Image Recognition and won the ImageNet 2015 Recognition and Detection challenges. We use this network for all our experiments on Low-shot dataset.

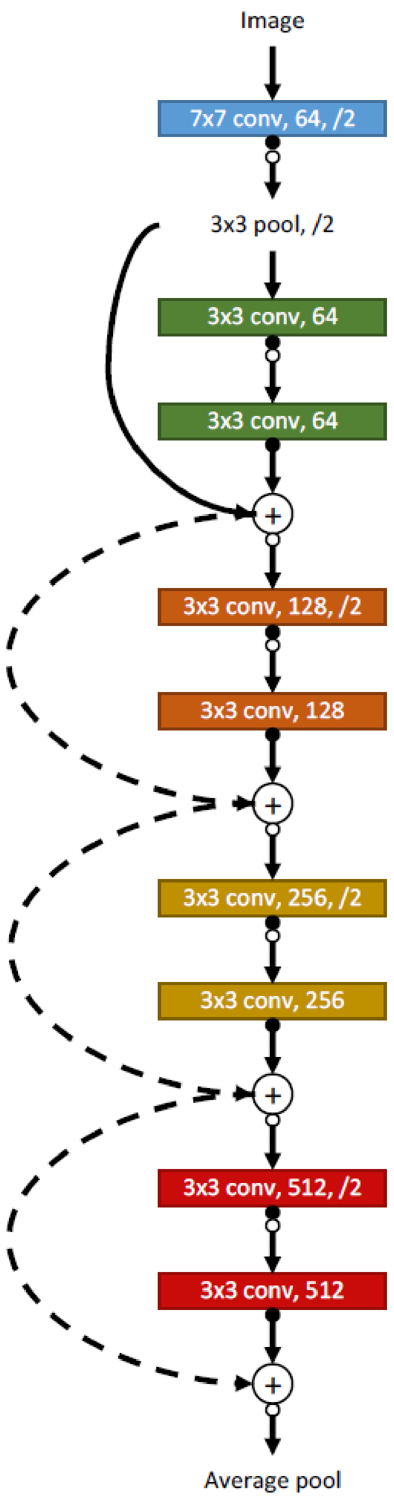


Figure 3. ResNet10 architecture. White circles represent ReLU and black circles show batch normalization. Image credit: Supplementary material [6]

Since we are using the ResNet model, we don't explicitly add dropout layers or regularization techniques because ResNet makes certain assumptions about the function to model that makes training very deep networks possible. It tries to find a solution close to identity mapping. At each stage it aims to model the residual by using stacked networks. The identity mapping is modelled by shortcut connections from the input to the output of each stage. The architecture of this network is described in Figure 3.

We feed images from our training set to this network using an in-built data loader. The model is trained using SGD to optimize cross entropy loss over the class labels. This gives us a test accuracy of 82% on our base classes and 32% on the novel classes, from which we concluded that the classifier performs well when there are sufficient training examples, but is significantly affected when this number goes down.

4.1.1 Data augmentation by jittering

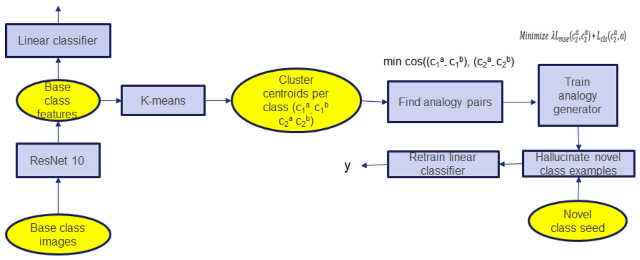
Following this, we implement a preliminary augmentation strategy based on jittering the pixels of input images to create new images. We experimented with four kinds of transforms: random rotation within a range of angles, random cropping and scaling within a boundary, horizontal flipping of images, and brightness transforms that increase and decrease the brightness. Of these, we found that the brightness transform resulted in images that were patchy and occluded, and decided not to include it in the final set of transforms used.

After applying the above transformations to our input pixels, we saved the generated images into our training set. Since we had 5 example images per novel category, each example resulted in over 8-10 jittered images, which made our novel set almost as big as our base set. However, the generated images were still fairly similar to the original ones and intra-class variations between the images wasn't large. We then re-trained our classifier on this new novel set, resulting in test accuracy going up by over 6 points on the novel set. This validated our hypothesis that augmentation improves performance, and we implemented a more targeted generation strategy, which is discussed below.

4.2. Low-shot learning by hallucination

The idea here is still to augment the training set with generated examples of the novel classes. However, the augmentation is done with respect to image features, and not raw image pixels anymore. In order to operate in this space, we can no longer train our ResNet end-to-end, and thus break it up into a feature extractor, which is trained in the representation learning phase, and a linear classifier, which is trained in the low-shot learning phase.

- Representation learning - Using the base categories alone, a feature extractor is trained. Our feature extractor is the ResNet-10 that we described above, however, [6] also experiment with ResNet-50, which they find performs better. Since our data set is quite small, we didn't want to increase the capacity of our model too much as it would overfit, and so we stick to ResNet-10. Another key difference from the original paper is that we currently use Cross-Entropy Loss on the class labels, whereas [6] use Squared Gradient Magnitude loss, which was formulated to encode the low-shot training objective. We wanted to experiment with the SGM loss as well, which gave them an 8 to 10 point improvement over vanilla classification loss, however, implementing it was tricky and we didn't have enough time.


Linear classifier
Base class features
ResNet 10
Base class images
K-means
Cluster centroids per class (c_1^a, c_1^b c_2^a c_2^b)
min cos((c_1^a- c_1^b), (c_2^a- c_2^b))
Find analogy pairs
Minimize $\lambda L_{mse}(c_2^a, c_2^a) + L_{cls}(c_2^a, a)$
Train analogy generator
Hallucinate novel class examples
Novel class seed
Retrain linear classifier
y
Figure 4. Pipeline of generation by hallucination

- Low-shot learning - In this phase, the learner is given a mix of features from both the base and novel classes, and has to learn a classifier to distinguish between them. For our dataset, this means a total of 14 class labels. In order to improve its performance on the novel class, the learner is allowed to add new examples back into the training set using the features given to it. Our learner does this by hallucination, which we describe below.

4.2.1 Hallucination

The intuition behind hallucination is to transfer transforms that cause significant intra-class variation from one class to another in order to generate examples. This can be broadly divided into four steps:

1. Clustering: The features of each base class are grouped into clusters using K-means clustering. The number of clusters is an important hyperparameter that had a huge impact on our final results. We used 10 clusters per class, and since each class has 50 images, that would result in 5 images per cluster. Each cluster is represented by a vector which is the cluster centroid. Although we aren't able to visualize the results of clustering, intuitively we expect the clusters to represent distinct variations; for instance, all front-facing face images in a class would go into one cluster, whereas profile images would go into another.

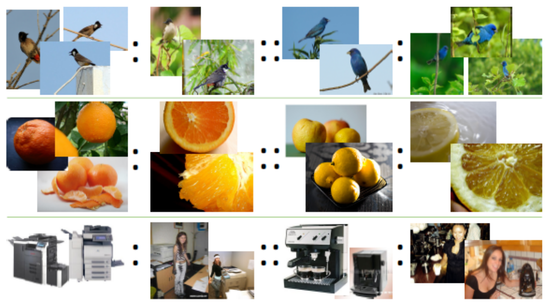


Figure 5. Example mined analogies. Each row shows the four image clusters that form the four elements in the analogy. Row 1: birds with a sky backdrop vs birds with greenery in the background. Row 2: whole fruits vs cut fruit. Row 3: machines (printer, coffee making) in isolation vs the same machine operated by a human.

2. Finding analogy pairs: Building on the intuition above, the difference between two cluster centroids c_1^a and c_2^a in the same class a can be considered as a transformation. For each pair of centroids within a class, we then search for an analogous pair of centroids in all other classes. This is done by finding a transform c_1^b - c_2^b in another class b such that the cosine distance between (c_1^a - c_2^a) and (c_1^b - c_2^b) is minimized. To best illustrate what an analogy pair would look like, we present an image from the Hariharan and Girshick paper (Fig. 5) although it's based on classes in ImageNet.
3. Training a generator: All analogy pairs found above with cosine similarity greater than zero are compiled into a dataset D_G. This is used to set the parameters of a generator G using a combination of regression and classification. From (c_1^a, c_2^a, c_1^b, c_2^b), we feed (c_1^a, c_1^b, c_2^b) as input to the generator and ask it to generate $\widehat{c_2^a}$. In addition to this, it has to classify $\widehat{c_2^a}$ into class a. The mean squared error between $\widehat{c_2^a}$ and c_2^a, called $L_{mse}(\widehat{c_2^a}, c_2^a)$ forms the regression loss, and is used to train the generator along with cross entropy loss on classification, called L_{cls}(W, $\widehat{c_2^a}$, a) (where W is the predicted class). Figure 6 shows how both losses decrease across epochs while training.
4. Generating examples: In order to generate examples for a novel class with label l, we feed a seed image from the existing novel class images into the trained generator, along with a pair of centroids c_1^a and c_2^a where the class a is chosen at random. The generator will then output a hallucinated feature vector which is added to the training set for the class l.

The number of examples to generate per novel class is also a hyperparameter. In the original paper, they determine this dynamically since each novel class has a different number of examples. But since all our classes have 5 examples

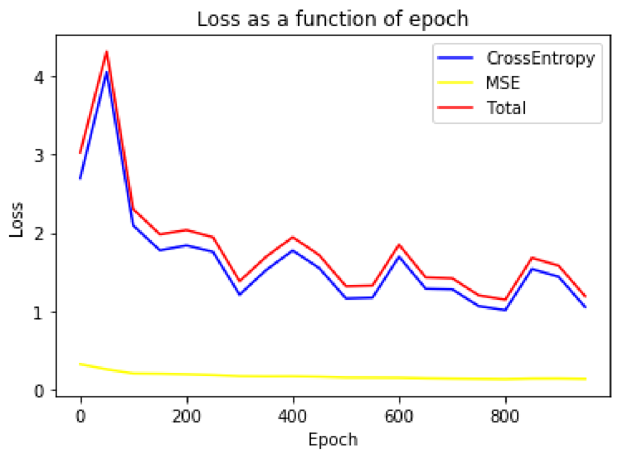


Figure 6. Loss when using extracted features for classification

each, we generate 5 more images per class. This results in a total of 10 images per novel class, which doubles the data available for the classifier to learn from. On the contrary, when we add too many hallucinated examples, the performance drops.

5. Experiments

5.1. Naive generation

We first describe the results of our preliminary approach in combination with the naive augmentation strategy. As described above, the baseline for this was a ResNet trained on base and novel classes together, and the results are in Table 1. Since our dataset is fairly small, we are able to achieve 100% training accuracy on the network within 10 epochs. We find that test accuracy is not so high and reaches 82% on the base classes. This could possibly be due to overfitting, but is high enough to convince us that it generalizes. As expected, while testing on novel classes, it doesn't do well and achieves 32%, less than half as on the base classes. We also report accuracy on base and novel classes together mainly to show the impact of augmentation.

Class	Train Accuracy(%)	Test Accuracy (%)
Base	100	82
Novel	100	32
Base + Novel	100	58.9

Table 1. Baseline results using ResNet10

After augmenting the training set by our naive jittering approach, we are interested in the test performance on the novel classes, since no new examples were added for the base classes. The results for this are reported in Table 2. We find that there is an increase in accuracy of over 6% on the novel classes alone, but only 2% when the base and novel

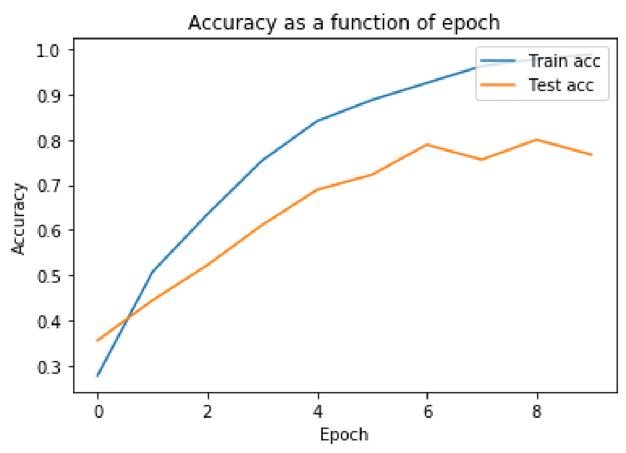


Figure 7. Jittering on novel classes

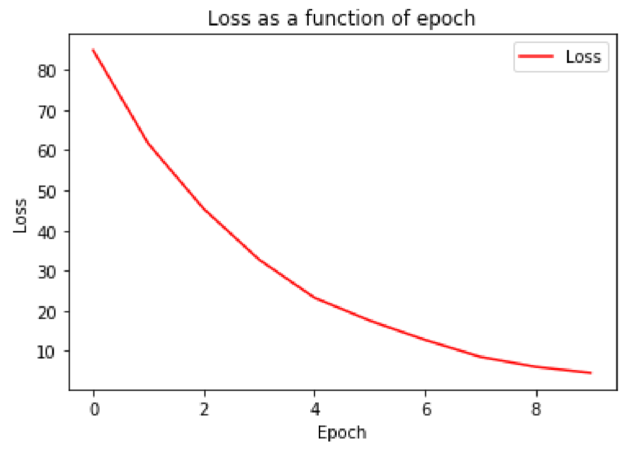


Figure 8. Jittering on novel classes

classes are tested together. Also, since this approach resulted in an almost 10-fold increase in training data, training accuracy drops a little, meaning the network can no longer memorize the data in 10 epochs.

Class	Train Accuracy(%)	Test Accuracy (%)
Novel	100	38.7
Base + Novel	98.89	60.43

Table 2. ResNet10 results after Jittering on novel classes

5.2. Generation by hallucination

We train a ResNet again as a feature extractor, but this is only trained on images from the base class. That is, the parameters of the feature extractor are set solely by looking at the base set, although during test time, the trained extractor is also used to extract features from the novel class. The features extracted are 512-dimensional, and are passed to a linear classifier that gives the probability distribution of an input image (specifically, image feature) over 14 classes. The extractor and the classifier are trained separately using SGD. To summarize, the feature extractor is trained on base class alone, while the classifier is trained on base and novel classes.

5.2.1 Baseline results

For learning the low-shot classifier, a learning rate of 0.1 is used. We use cross entropy loss and stochastic gradient descent with momentum 0.9. The weight decay is set to 0.0001. The results are recorded in Table 3. Interestingly, on testing the classifier with features from the base class test set, we get an accuracy of 72%, which is 10 points lower than the end-to-end trained ResNet. This could be because training the classifier separately, without giving it access to the parameters of the convolutional layers or the input image pixels is resulting in loss of information. Hyperparameter tuning did not majorly improve this number. On the novel classes alone, as well as on base and novel classes together, the classifier achieves a higher test performance than the end-to-end ResNet did.

Class	Train Accuracy(%)	Test Accuracy (%)
Base	100	72
Novel	100	36.43
Base + Novel	100	59.2

Table 3. Classification using extracted features

5.2.2 Generation results

Cluster size	Max examples per class	Test Accuracy (%)
10	10	40.8
5	10	28.5
5	20	24.7

Table 4. Novel class accuracy with different hyperparameters

As mentioned in the Section 4.2, the most important hyperparameters for our generation approach was cluster size and maximum number of examples to generate per class. We used a cluster size of 10 ([6] use 100 clusters per class for ImageNet). Also, we generated until there were 10 examples per class, and since we already have 5, there were 5 additionally generated images for each class. The effect of other hyperparameters is tabulated in Table 4.

Class	Train Accuracy(%)	Test Accuracy (%)
Novel	100	40.8
Base+Novel	100	62.14

Table 5. Classification using generated examples

We then re-trained the classifier on three kinds of features: features from the base set, features from the small novel set, and hallucinated features for the novel set. We then measured test set accuracy on the novel class test features alone, as well as base and novel test features together. The results are described in Table 5, where it can be seen that we achieve a 4% improvement over our baseline.

6. Conclusions and Future Work

In this project, we implemented the ResNet10 architecture to obtain a baseline for the Face recognition classification task. We explored the use of jittering as a technique to augment our dataset which performed better than the baseline model. Following this, to improve upon this baseline, we extract features from our base classes and implemented the hallucination technique as proposed by Hariharan and Girshick [6] to augment our novel set. This resulted in a 4% improvement over our baseline. While not a dramatic increase, it convinces us that augmentation through hallucination has potential in improving tasks where training data is sparse. In the future, we would like to explore stronger feature extractures for this domain, such as VGG-Face to see if they work better than the ResNet10 architecture used in this project. Another possible direction is to explore the use of larger datasets with more number of classes and cleaner, better aligned images than the one used in this project.

References

[1] T. Ahonen, A. Hadid, and M. Pietikäinen. Face recognition with local binary patterns. In *European conference on computer vision*, pages 469–481. Springer, 2004.

[2] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. *CoRR*, abs/1606.05233, 2016.

[3] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun. A practical transfer learning algorithm for face verification. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3208–3215. IEEE, 2013.

[4] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *European Conference on Computer Vision*, pages 566–579. Springer, 2012.

[5] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. MS-Celeb-1M: A dataset and benchmark for large scale face recognition. In *European Conference on Computer Vision*. Springer, 2016.

[6] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proc. of IEEE Int. Conf. on Computer Vision (ICCV), Venice, Italy*, 2017.

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

[8] G. B. Huang, H. Lee, and E. Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2518–2525. IEEE, 2012.

[9] 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.

[10] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[11] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, June 2015.

[12] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[13] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Web-scale training for face identification. *CoRR*, abs/1406.5266, 2014.

[14] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[15] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[16] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[17] Y. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision (ECCV)*, October 2016.

[18] W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets, and J. Weng. Discriminant analysis of principal components for face recognition. In *Face Recognition*, pages 73–85. Springer, 1998.