Yearbook Age Prediction (CS395T)

Darshan Thaker
The University of Texas at Austin
dbthaker@cs.utexas.edu

Abstract

We tackle the yearbook 'Face To Age' challenge to predict the year that a given yearbook photo was taken in. We experiment with three variants of the VGG19 architecture [4], the AlexNet architecture [2], as well as different loss functions and an extensive effort to reduce overfitting. A variant of the AlexNet architecture, that has 2048 *neurons in the fully connected layers and is ensembled over 5 models, gives us the best results, with a mean L1 norm distance of* 5.54*.*

1. Introduction

The task of this project is to use deep convolutional neural networks to determine what year certain yearbook photos were taken in. The dataset consists of 22840 training images, which are frontal-facing $171 \times 186 \times 3$ images. For this task, we use two base network architectures pretrained on ImageNet: VGG19 [4] and AlexNet [2]. These architectures perform very well on ImageNet and have shown good generalization results in various domains [3]. We fine-tune the fully connected layers of these architectures on the given training dataset. The variants of these architectures explored include changing the number of neurons in the fully connected layers and adding additional fully connected layers. We find that the architecture that performs best is an ensemble of 5 AlexNet models.

The report is organized as follows: Section 2 describes the technical details of the input modification, various architectures tried, loss functions, model hyperparameters, and an extensive discussion of preventing overfitting. Section 3 discusses the experiments tried to evaluate the different architectures and design decisions as well as gives examples of wins and losses for the model.

2. Methodology

2.1. Input Preprocessing

We pose the problem as a classification problem across the 104 labels in the training set. Consequently, we encode labels as a one-hot vector of size 104. This is because the year labels are not equidistant, so if the network were to predict year integers, there would be an inherent distance imposed by different labels. Instead, we treat labels as categorical values, which motivates a one-hot encoding. We also explore smoothing one-hot labels. For example for the ith sorted label, a smoothed one-hot encoding might have 1 in the ith location of the one-hot encoding, and a 0.5 in the $i-1$ and $i+1$ location in the one-hot encoding. This would help the loss function penalize less for predictions that are very close to the original label in terms of year distance.

2.2. Architectures

The first base model architecture used for experiments is the 19-layer VGG19 architecture. VGG19 has an input layer followed by 16 convolutional layers, 2 fully connected layers, and an output layer. The convolutional layers are divided into 5 blocks and each block is followed by a spatial max-pooling operation. We explore three variants of this model architecture.

The first variant is the vanilla VGG19 architecture with one additional fully connected layer, consisting of the base architecture connected to four fully connected layers, where the number of neurons in the last fully connected layer is equal to the number of classes. The weights for these fully connected layers are initialized using the random Xavier initialization. The second variant is the vanilla VGG19 architecture. The third variant has two branches of three fully connected layers coming from the last pooling layer in the base VGG19 architecture. The goal of this third architecture is to force one branch to learn features for predicting years and another branch to learn features for predicting the decade of an image. This is accomplished by optimizing one branch on a loss function based on the year labels, and the other branch optimized on a loss function based on a more general decade labels. For all these variants, we optimize the parameters of all the fully connected layers.

The second base model architecture explored is the 5-layer AlexNet architecture. The only variant experimented

is the 5-layer architecture followed by 3 fully connected layers. The final fully connected layer has the number of neurons equal to the number of classes.

We take the number of neurons in each fully connected layer in both models as a hyperparameter that we search over to find the optimal performance on the validation set.

2.3. Loss functions

The loss function used for training is the categorical cross entropy loss between the output of the neural network and the one-hot labels added to a L1 regularization loss term with coefficient 1e-6. Formally, let X denote the output of the neural network, let Y be the $m \times l$ matrix of one-hot labels where l is the number of classes and m the number of examples, and let $H(\cdot)$ denote the cross-entropy function. Then, the loss is defined as

$$L_w(X, Y) = H(X, Y) + \|w\|_1 \quad (1)$$

The alternative loss function explored is the loss function for the 3rd variation of the VGG architecture, which uses a classifier for identifying the decade of an image. In this case, the loss is the cross entropy between the matrix of one-hot labels of decades for training images and the output of the decade branch of the neural network. The total loss of the network is the loss of the decade branch plus the loss of the year branch (same as Equation 1).

2.4. Model Hyperparameters

As with most neural networks, there are many hyperparameters that need to be tuned. This section summarizes the list of hyperparameters tried and the final choice of these parameters for the best-performing model. The effectiveness of hyperparameters was determined empirically by observing how the loss function drops during training as well as how training and validation accuracy are affected during training. For training, we use the Adam optimizer with learning rate 0.001 that is decayed to 0.0001 and 0.00001 after 100000 and 300000 iterations respectively. The dataset is processed in batch sizes of 32 and run for 10 epochs, where in each epoch, a pass over the entire dataset is made in shuffled random order. We experimented with adding more epochs, but the training loss was very low after 10 epochs, and the validation accuracy was unaffected by more iterations. We kept the number of neurons in the fully connected layers as a hyperparameter, typically ranging over 1024, 2048, and 4096 neurons.

2.5. Overfitting

A substantial amount of time was spent reducing overfitting. We observed that after one or two epochs, the training L1 norm distance and loss were near 0, but the validation L1 norm distance was very high, which is a classic sign of overfitting. To reduce overfitting, we employed 7 different techniques: dropout layers, batch normalization layers, weight regularization, data augmentation, training ensembles, and early stoppping.

In the fully connected layers near the end of all architectures, we follow each fully connected layer with a *batch normalization* layer [1] and a *dropout* layer [5] that drops neurons with probability 0.5 . These have been shown empirically to substantially reduce overfitting in neural networks. In practice, networks with batch normalization are more robust to bad initialization. Additionally, we add *L2 weight regularization* to all fully connected layers. This is helpful in reducing overfitting since it penalizes weight vectors that are large in one dimension but small in one dimension. We do not desire such lopsided weight vectors since that does not make use of all the input, and would weigh some inputs much more than others, reducing overall generalization. Furthermore, adding *data augmentation* has been shown to reduce overfitting. Specifically, we augment the training data with random mirroring, random brightness changes with a max delta of 0.2, random saturation changes with a factor from 0.3 to 1.3, and random hue changes with max delta of 0.2. Data augmentation helps the network generalize better since it learns robust features in spite of changes in orientation, brightness, saturation, and hue. We did not explore adding random rotations to the training data since the training and validation data photos typically obey the same orientations, so we hypothesize that adding random rotations of more than 10 degrees would not help generalization performance. We added *ensembles* of models, which reduced overfitting the most out of the above techniques. Ensembling is useful because of different random network initializations. Averaging the network output over multiple models typically yields more generalizable networks. We could not add ensembles of our VGG architectures because of memory constraints on GPUs, but we explored adding ensembles of AlexNet, which improved our generalization error by reducing validation accuracy. Finally, over 10 epochs, we evaluated validation L1 norm distance after every epoch, and picked the model with the best validation error, a form of *early stopping*.

3. Experiments

There are various differences in VGG19 and AlexNet, mostly in complexity of the model. Although VGG19 is much more complex than AlexNet and theoretically should be able to capture more complex intricacies in the data, we find that AlexNet outperforms VGG19 in terms of validation accuracy. We hypothesize that the VGG19 has many layers that are pretrained on ImageNet, and we do not train on these fixed weights. Since we only finetune the fully

(a) 1974 (predicted) vs. 2005 (actual)

(b) 2008 (predicted) vs. 1979 (actual)

(c) 1965 (predicted) vs. 1965(actual)

(d) 2001 (predicted) vs. 2001(actual)

Figure 1: Visualized Wins and Losses

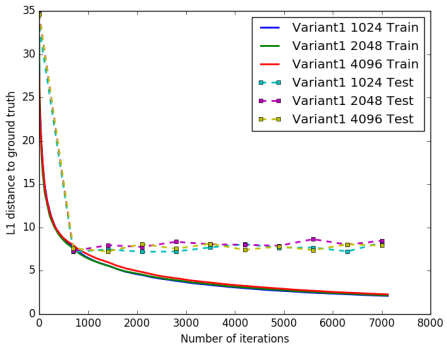

Figure 2: VGG Variant 1- Train/Test L1 Distances

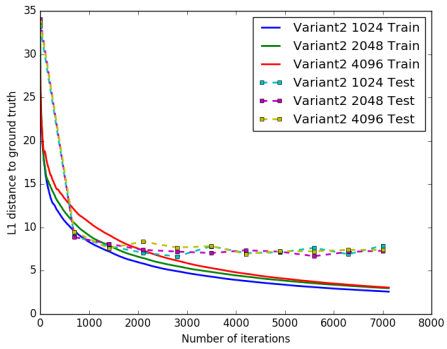

Figure 3: VGG Variant 2 - Train/Test L1 Distances

connected layers, it may be the case that the previous layers have low transfer capabilities to the Yearbook dataset because of high complexity features. For this project, we did not explore finetuning a few of the convolutional layers because of the lack of large amounts of training data, but this would help improve accuracy of VGG19 on the Yearbook dataset. Recall that we experimented with 3 variants of VGG19, as discussed in Section 2.

We now discuss training and testing error for the variants of the architectures tried, for both VGG19 and AlexNet. In Figures 2, 3, 4, 5, 6, the training error is smoothed every 5 iterations to obtain a more consistent and interpretable plot. Since the optimizer is trained with a minibatch of 32, the gradients (and consequently the parameter) updates are noisy, leading to very jittery error plots. Thus, we replace the error at each iteration with the mean of the 5 errors in preceding iterations to perform an effective smoothing of the error plots. Additionally, the test errors are calculated at the end of every epoch. In each epoch, we iterate through the entire training dataset in minibatch sizes of 32, which is

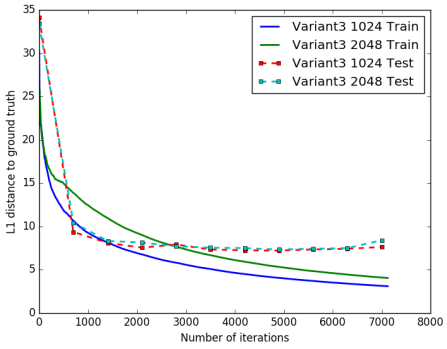

Figure 4: VGG Variant 3- Train/Test L1 Distances

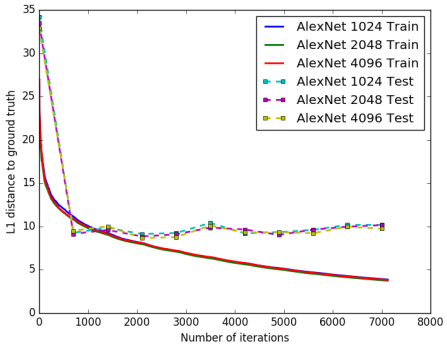

Figure 5: AlexNet - Train/Test L1 Distances

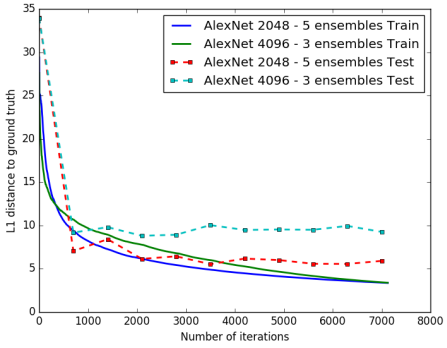

Figure 6: AlexNet Ensembles - Train/Test L1 Distances

equivalent to about 700 gradient descent steps per epoch.

Looking at Figure 2 we see that the number of neurons in the fully connected layers did not affect generalization error or training error that much, suggesting that accuracy had already saturated with a low number of neurons. This is expected since the first variant adds one more fully connected layer to the vanilla VGG19 architecture, which introduces many more parameters. Given the low amount of training data, it is expected that the training error and generalization error will not be affected too much by varying the number of neurons in the fully connected layers.

However, in Figure 3, we see that the vanilla VGG19 architecture is more distinctly affected by the number of neurons in the hidden layer. Here, we see an inverse relationship between the training and testing L1 distance to ground truth. Specifically, as we increase the number of neurons in the fully connected layers, the training error is higher overall. This intuitively makes sense, since we are learning many more parameters with the same amount of training data. Furthermore, we observe that on average, validation accuracy is higher for the vanilla VGG19 architecture with fully connected layers that have fewer number of neurons. This suggests that if we add more neurons to the fully connected layers, we are overfitting to the training data. This is corroborated by the number of trainable parameters in the network as proportion to the number of input data, which is much larger than the number of training examples with 4096 neurons in the fully connected layers.

In Figure 4, we see the decade architecture variant of VGG19 that optimizes for both year and decade losses. We only explore number of neurons to be 1024 and 2048 and not 4096 because of memory limits on GPU. This architecture is special in that it exhibits similar average validation error to the other two variants, but overfits much less. This suggests that the decade loss helps the model generalize better.

The AlexNet architectures perform better on average than the VGG19 architectures, which can be seen in Figure 5 and Figure 6. Observing these figures also shows the concrete difference of ensembling on overfitting. Specifically, in Figure 6, we obtain the model that performs best in terms of validation L1 distance to ground truth labels. In epoch 4, the model has average validation L1 distance of 5.54, which outperforms all the other model variants.

Fixing the AlexNet 5-ensemble with 2048 neurons per fully connected layer, we turn to some qualitative analysis of how the model performs on the validation set. In Figure 1, we see 2 wins and 2 losses of this neural network. It is difficult to reason about why the model misclassified the two photos in (a) and (b) by purely looking at the images, since the task of classifying yearbook photos is quite challenging even for a human. However, in (a), we hypothesize that the artifacts in the image besides the face that might have confused the neural network. Additionally, it seems that image quality is a big factor in predicting year, since (a) and (c) seem to have similar quality of images, suggesting a older year. In contrast, (b) and (d) look more recent to the human eye.

4. Conclusion

We have demonstrated a deep convolutional neural network based on an ensemble of the AlexNet architecture that gives as low as 5.54 mean L1 error on the validation set. We explored various ways to prevent overfitting and trained various variants of models. A future direction might involve ensembling over different variants itself. For example, we observed that the decade variant of the VGG19 architecture has high resistance to overfitting. It is possible that an architecture that ensembles over the 3 variants discussed might perform very well.

References

[1] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[3] H. C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, May 2016.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[5] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.