CS337 Course Project
Face Recognition

Pranjal Kushwaha, Shashwat Garg, Vedang Asgaonkar, Virendra Kabra

Autumn 2022

Contents

1 Dataset

- We use the Labelled Faces in the Wild (LFW) dataset. It has been taken from Kaggle. There is an uneven distribution of images, with only 10 people having at least 53 images. Thus, we train the model to classify these 10 people, with our dataset containing 53 images for each.
- The train-test split is 80-20, and the train set is further split to create the validation set. Equal splits are created for each class.
- **Data Augmentation**: We add horizontally flipped images to the dataset. This acts as a regularizer, helping the model to generalize better.

2 Model Architecture

We use a Convolutional Neural Network (CNN) to create a multi-class image classifier. The network is inspired from FaceNet [?].

Layer	In	Out	Kernel	Params
conv1	$250 \times 250 \times 3$	$123 \times 123 \times 64$	$7 \times 7 \times 3, 2$	9K
batchnorm1	$123 \times 123 \times 64$	$123 \times 123 \times 64$		128
relu1	$123 \times 123 \times 64$	$123 \times 123 \times 64$		0
maxpool1	$123 \times 123 \times 64$	$61 \times 61 \times 64$	$2 \times 2 \times 64, 2$	0
dropout1	$61 \times 61 \times 64$	$61 \times 61 \times 64$		0
conv2	$61 \times 61 \times 64$	$61 \times 61 \times 128$	$3 \times 3 \times 64, 1$	74K
batchnorm2	$61 \times 61 \times 128$	$61 \times 61 \times 128$		256
relu2	$61 \times 61 \times 128$	$61 \times 61 \times 128$		0
maxpool2	$61 \times 61 \times 128$	$30 \times 30 \times 128$	$2 \times 2 \times 128, 2$	0
dropout2	$30 \times 30 \times 128$	$30 \times 30 \times 128$		0
conv3	$30 \times 30 \times 128$	$30 \times 30 \times 256$	$3 \times 3 \times 128, 1$	295K
batchnorm3	$30 \times 30 \times 256$	$30 \times 30 \times 256$		512
relu3	$30 \times 30 \times 256$	$30 \times 30 \times 256$		0
maxpool3	$30 \times 30 \times 256$	$15 \times 15 \times 256$	$2 \times 2 \times 128, 2$	0
dropout3	$15 \times 15 \times 256$	$15 \times 15 \times 256$		0
conv4	$15 \times 15 \times 256$	$15 \times 15 \times 64$	$3 \times 3 \times 256, 1$	148K
batchnorm4	$15 \times 15 \times 64$	$15 \times 15 \times 64$		128
relu4	$15 \times 15 \times 64$	$15 \times 15 \times 64$		0
dropout4	$15 \times 15 \times 64$	$15 \times 15 \times 64$		0
flatten	$15 \times 15 \times 64$	14400		0
fc1	14400	1024		14M
relu5	1024	1024		0
dropout5	1024	1024		0
fc2	1024	64		66K
relu5	64	64		0
dropout5	64	64		0
fc3	64	10		650
Total				15M

Table 1: Model Architecture

- Convolutions: To learn hierarchical representations of the input data, we use several convolutional layers.
- Batch Normalization: Adding these layers lead to faster convergence.
- ReLU: These are added for non-linearity, which is necessary for the universal approximation theorem to hold.
- Max Pooling: This helps make the representation become approximately invariant to small translations of the input.

- Dropout: This acts as a regularizer. We use dropout with $p = 0.2$ after the maxpool layers [?], and with $p = 0.5$ after the fully-connected layers [?].
- Optimizer: Stochastic Gradient Descent (SGD) with a learning rate of 10^{-3} and weight decay of 10^{-3}.
- Loss: We use two losses in addition:
  - Cross Entropy Loss: This is the standard loss for multi-class classification.
  - Triplet Loss: This is introduced in [?]. We view the neural network as having an encoder and a decoder. The neural network until its penultimate layer is the encoder, which produces a latent representation for each image. The final layer acts as a decoder on this. This loss encourages clustering in the latent representation.

$$L(a,p,n) = \sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

    where a is the anchor, p is a positive sample (having same class as a), n is a negative sample, and α is the margin (1 is used).

3 Analysis

- Batch Size

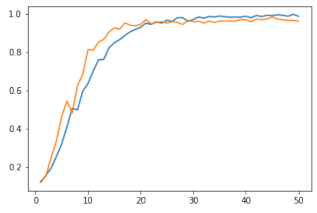
Figure 1: Batch Size 1

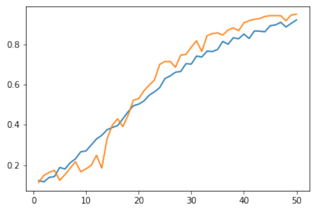
Figure 2: Batch Size 5

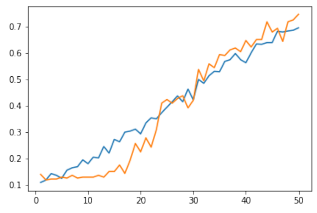
Figure 3: Batch Size 10

A batch size of 1 leads to better and faster convergence.

- Batch Normalization

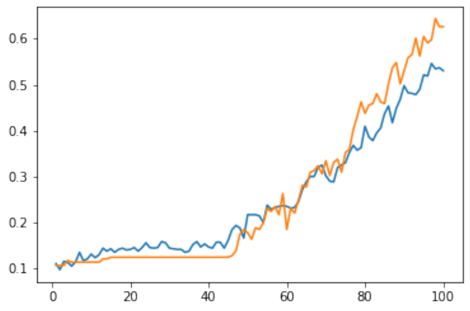
Figure 4: Without Batch Norm

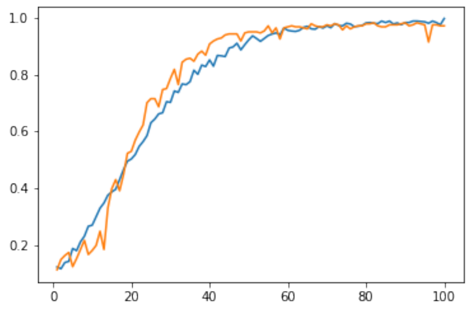
Figure 5: With Batch Norm

Faster convergence is observed with batch normalization.

- Optimizer

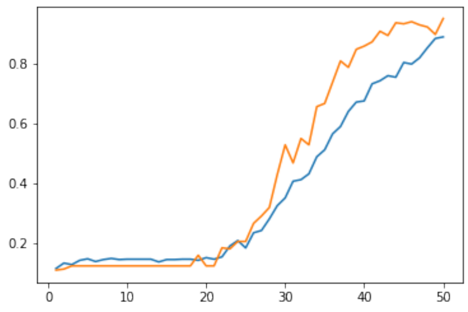

Figure 6: Adam

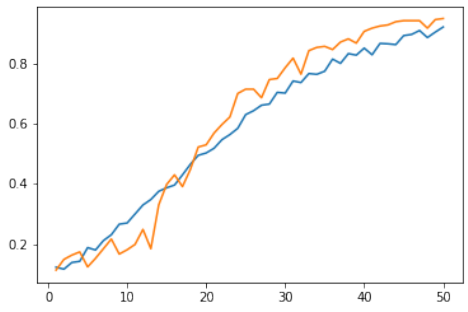

Figure 7: SGD

4 Interpretable Architecture

Alongside the categorical cross entropy loss, we use the triplet loss as introduced in FaceNet [?]. We view the neural network as an encoder-decoder combination. The network upto its penultimate layer is the encoder and the last layer is the decoder. Thus the encoder produces a 64 dimensional latent representation of each image. The triplet loss encourages clustering in this latent space i.e. the latent representations of images in the same class are close together.

The triplet loss L_T defined on a batch of latent vectors F is given by

$$L_T(F) = \sum_{f_a \in F} \sum_{f_p \in F_p(f_a)} \sum_{f_n \in F_n(f_a)} [||f_a - f_p||^2 - ||f_a - f_n||^2 + \alpha]_+$$

where $F_p(f_a)$ is the subset of F having the same class as f_a, $F_n(f_a)$ is its complement and α here is margin which we set to 1.

Thus, the loss forces latent vectors of the same class to reduce their Euclidean distance, while increasing the Euclidean distance between vectors of different classes.

By employing the triplet loss, the latent vectors get clustered, as measured by the ratio of mean squared distance within a cluster to the mean squared distance between clusters. This ratio goes down to roughly 0.3 by employing the triplet loss.

5 Adversarial attack on the CNN

We demonstrate an attack on the CNN, in which we provide images which are visually similar to the original image, but get misclassified. To that end, we train a generator which is designed to fool the model. The generator adds a bias to each pixel of the image. We first freeze the model weights. The output of the generator is passed to the model as input. We train this composite system using a loss which is negative of the cross entropy loss. This encourages the generator to produce images which are misclassified. We also add a regularizer, to encourage similarity of the generated image with the original one.

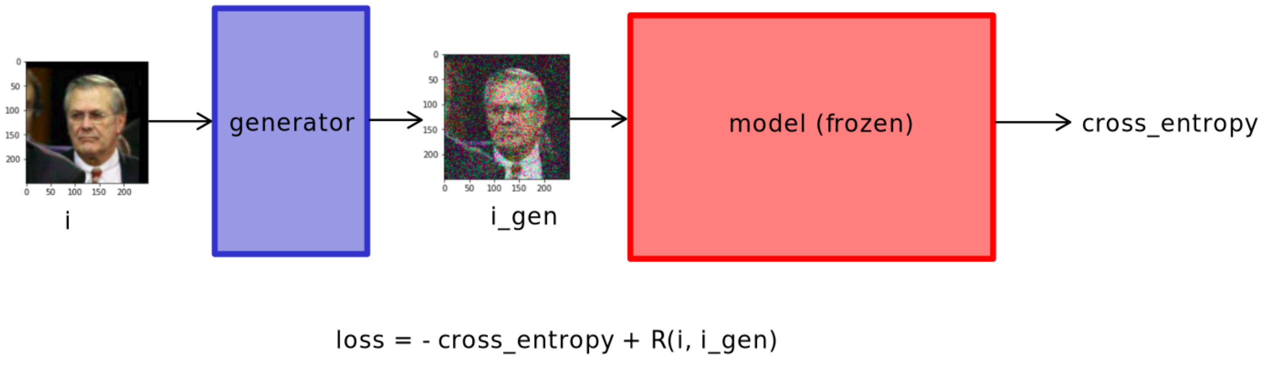

Figure 8: Adversarial attack

Observations:

- By adding a l_2 regularizer with $\lambda = 20000$, the generator brings down the model's test accuracy to 35% with minimal loss of visual similarity.
- l_1 regularizer does not do perform as good as l_2. Trained with $\lambda = 100$, it brings down the test accuracy to 38% but ends up adding too much noise.

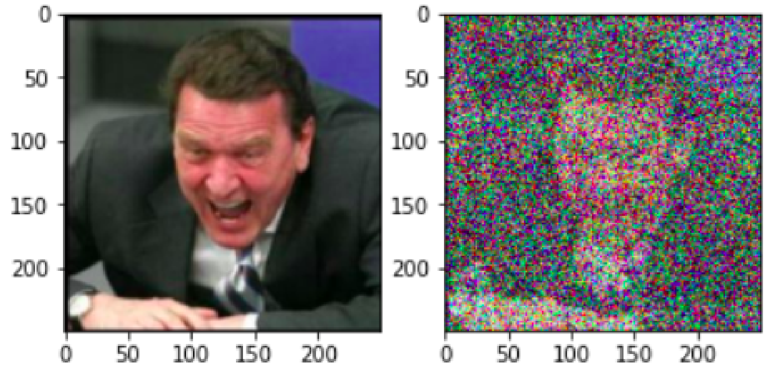

Figure 9: l1 regularizer

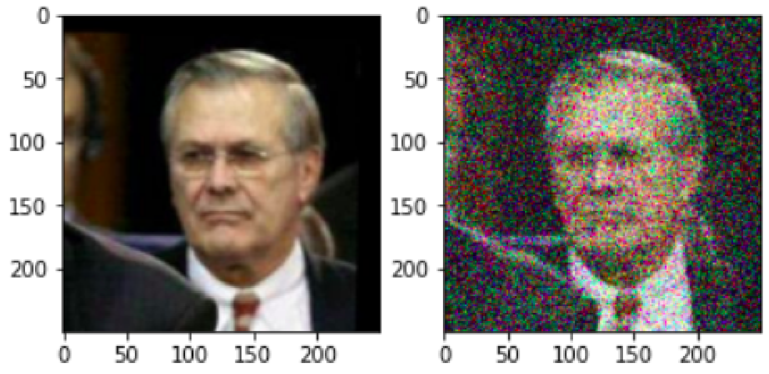

Figure 10: l2 regularizer