FACE RECOGNITION WITH CNN USING A CELEBRITY PINS DATASET

Kushal Chaudhari, Karthik Narasimhan, and Bahram Khodabandehlouee

***Index Terms*—face recognition, convolutional neural network, CNN, deep learning, dropout layers**

I. Abstract

Face recognition has been one of the important algorithms to have been developed in the area of artificial intelligence. It is of critical importance as it is something that we are able to do effortlessly since birth. It acts as a good test for the accuracy of a computing system in its given classification task using AI models. The research surrounding the same has become one of the most prominent areas in the domain of computer vision and machine learning. Convolutional Neural Networks are a class of artificial neural networks that has been shown to be highly effective in multi-class image classification problems like face recognition. We trained a Convolutional Neural Network (CNN) architecture to measure performance on training and validation sets for our dataset. We have compared and contrasted the performance of our neural network by tuning various hyperparameters such as weight initializations, learning rates, kernel size, strides, mini-batches, epochs, using dropout layers to aim for better generalization.

II. Introduction

Today, a lot of applications in machine learning have demanded an analysis that a person's face provides. This includes information about their identity, expression, gaze, health and age. As per research in neuroscience, the process of face recognition involves a structural encoding and a comparison with mental images of faces which activate multiple brain regions including the middle temporal lobe, hippocampus, parahippocampal gyri, amygdala, inferior frontal gyrus, and orbitofrontal cortex. [1] This understanding of how neural networks in the human brain work has led to the development of artificial neural networks (ANNs) that can recognize faces from photos, and videos. Face recognition is essentially an image classification problem. The most popular class of ANNs that perform well on this problem are the convolutional neural networks (CNN). This paper evaluates and compares the accuracy of multiple CNN architectures and variations of hyperparameters built with only Numpy and Scipy while using OpenCV for pre-processing. Models are built from scratch without the aid of any additional frameworks such as TensorFlow or Torch.

With the presence of applications like Netflix and Amazon Prime, the research in this domain has been accelerated. Particularly, there is a need to automatically tag metadata for content on those platforms to help with data analysis and organization across their users and personalize recommendations. With platforms like Amazon Fresh, there is a need to track a particular customer through a store so as to match them with their account for seamless processing. The goal therefore is to train a neural network from scratch on a CNN architecture to be able to get good accuracies for the face recognition model from scratch.

III. Convolutional Neural Networks

CNNs must at least consist of one or more convolutional layers, one or more pooling layers, one or more fully connected layers, one or more activation layers and an objective function. In the convolutional layer, a convolution function is performed with the input data and the kernel, which is a small matrix used for blurring, sharpening or other image processing tasks. The output of the convolution function is called the feature map. It can be defined according to the following formula for a 2D image I and a 2D kernel K: [2]

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,\ j-n)$$

Machine learning libraries instead implement a related function called cross-correlation, which is the same as convolution but without flipping the kernel, as defined by the formula:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m,\ j+n)K(m,n)$$

In the pooling layer, the max pooling operation reports the maximum output within a rectangular neighborhood. Pooling helps to make the representation approximately invariant to small translations of the input. [2]

IV. Background and Related Work

A formal method of classifying faces was first proposed in [3] by Francis Galton. Galton proposed collecting facial profiles as curves, finding their norm, and then classifying other profiles by their deviations from the norm. This classification is multimodal, i.e. resulting in a vector of independent measures that could be compared with other vectors in a database.

Yann LeCun first proposed CNNs for image classification in [4] with the central premise that CNNs combine three architectural ideas to ensure shift and distortion invariance: local receptive fields, shared weights and spatial or temporal subsampling.

Within the specific research of CNNs for face recognition, researchers in [5] proposed the Squeeze-and-Excitation (SE) block and three improved structures via a channel attention mechanism: the depthwise SE module (DSE), the depthwise separable SE (DSSE) module and the linear SE (LSE) module. This research is relevant to this paper as our CNN was intentionally designed to be lightweight as our use of Numpy and Scipy only cause an extremely deep CNN with many layers and parameters to have an extremely long training time. Researchers attained a validation accuracy of 99.57% after testing their LSE model on the Labeled Faces in the Wild (LFW) compared to a training accuracy of 92.37%.

Researchers in [6] proposed Sparse Approximation as a method of attaining a high performance in face recognition with CNNs with very limited data. Researchers attained a validation accuracy of 84.86% after training a pre-trained CNN on the LFW dataset with just one image per class and 96.14% after training a CNN on the LFW dataset with thirteen images per class.

The creators of the CelebA dataset wrote a paper - *From Facial Parts Responses to Face Detection: A Deep Learning Approach*. [7] In the paper they have considered finding faces from a new perspective through scoring facial parts responses by their spatial structure and arrangement. This approach helped them achieve a high recall rate of 90.99% on the FDDB benchmark which also allows their network to detect faces under severe occlusion and unconstrained pose variation, which are the main difficulties and bottlenecks of most existing face detection approaches. In our implementation, we created a CNN and monitored the effect of altering hyperparameters in accuracies of training and validation sets to get the best result.

V. Applications

Different types of face recognition problems can be defined as: when we are given an image or even a video image taken from a scene of a movie and we will compare them with other images stored in a database to identify them.

The process of face recognition is a biometric approach that engages automated methods to investigate or recognize the identity of a living person based on phenotypic characteristics. Generally speaking, a biometric identification system makes use of either phenotypic features (such as a fingerprint, iris pattern of eyes, or face) or characteristic patterns (such as hand-writing, voice) to recognize a person. Because of human inherent protectiveness of his/her eyes, some people are reluctant to use eye identification systems. Face recognition has the benefit of being a passive, non-intrusive system to verify personal identity in a "natural" and friendly way [8] . Hence, face recognition process starts with the detection of face features in some cases with clutter backgrounds, proceeds by normalizing the face images to account for geometrical changes, possibly using information about the location and appearance of facial landmarks, identifies the faces using appropriate classification algorithms, and post processes the results using model-based schemes and logistic feedback [9] . By going over the literature, it becomes quite difficult to state that at the time, there exists a system that works perfect which solved all face detection problems with all fields of variations. Among all methods, neural networks have shown very good results for detecting a certain pattern in a given image [10], [7] .

VI. Approach & Results

Initially the images were already organized in a folder for the given dataset under different celebrities. From that, we chose 10 random celebrities (classes) to work with. The preprocessing for all the images in classes was done with the help of OpenCV and glob modules. Essentially, all the images were converted into grayscale and resized in 35 by 35 pixel dimensions to speed up the model and stored inside their respective class folders. All the images were then converted into 2D arrays across all the n number of observations to get a feature-set matrix of $X(n\times35\times35)$ along with their respective labels in matrix form $Y(n \times 1)$. These numpy matrices were saved to avoid preprocessing every time and were directly imported to run the models. For running the architecture, the dataset was split 2/3 for training and 1/3 for testing. After which the Y labels were converted into one hot encoding format to facilitate cross-entropy calculations. In some parts, the X and Ytraining data was split further into mini-batches of mostly 30 observations(image array representations) per batch. The accuracies through training and test sets were tested per epoch while varying different hyperparameters to produce J vs Epoch plots for the models. The X and Y representations were also randomly shuffled before forward and backward propagation.

We implemented various architectures with different values for hyperparameters such as learning rate, epochs and weight initialization, stride for max-pooling layer and the kernel window size. Generally, our implementation was in two categories of Stochastic Mini-Batch Gradient Descent and full batch Gradient Descent. In addition, due to getting high training accuracies and lower accuracies in validation in each set of experiments we implemented dropping out layers to observe the effect of it on better generalization to overcome overfitting in our data. The results for different models are summarized below:

It is noticeable that using mini batches could improve the accuracy by very low. Also, due to significant overfitting, we implemented dropping out layers to overcome this problem by inducing some regularization. Using drop out layers helped to improve the accuracy slightly, but not significantly. In file c.py we also experimented with L2 regularization but were not able to produce significant results.

TABLE I
CL: CONVOLUTIONAL LAYER, MPL: MAXPOOL LAYER, FL: FLATTENING LAYER, FCL:FULLYCONNECTED LAYER, SIG: SIGMOID ACTIVATION FUNCTION, CE: CROSS ENTROPY OBJECTIVE FUNCTION, SM: SOFTMAX ACTIVATION FUNCTION, DO: DROPOUT LAYER, TANH: TANGENT HYPERBOLIC LAYER, RELU: RELU ACTIVATION FUNCTION, LL: LOG LOSS OBJECTIVE FUNCTION

ARCHITECTURE	η	EPOCH	FCL INIT RANGE	DROP OUT LAYER (Y/N)	MINI BATCH (Y/N)	TRAIN ACC	TEST ACC
MODEL 1 CL(5) - MPL(3,2) - FL - FCL(289,50) - ReLu - FCL(50,10)-SM-CE	.05	75	[-0.067,0.067]	N	N	100%	42.26%
MODEL 2 CL(5) - MPL(3,2) - FL - FCL(289,50) - ReLu - FCL(50,10) - SM - CE	0.017	75	[-0.067,0.067]	N	N	96.26%	42.85%
MODEL 3 CL(5) - MPL(3,2) - FL - FCL(289,10) - SM - CE	0.2	20	[-0.067,0.067]	N	Y 30/batch	78.06%	43.15
MODEL 4 CL(5) - MPL(5,2) - FL - FCL(256,10) - SM - CE	0.05	50	[-0.067,0.067]	N	N	62.8%	45.15%
MODEL 5 CL(5) - MPL(5,2) - FL - FCL(289,100) - Tanh() - DO(0.6) - FCL(100,10) - SM - CE	0.003	75	[-0.067,0.067]	Y	Y 30/batch	60.59%	37.5%
MODEL 6 CL(5) - MPL(5,2) - FL - FCL(256,10) - SM - CE	0.03	75	[-0.067,0.067]	N	N	62.3%	43.4%
MODEL 7 CL(5) - MPL(5,2) - FL - FCL(256,10) - SM - CE	0.1	75	[-0.1,0.1]	N	N	74.18%	41.96%
MODEL 8 CL(5) - MPL(5,2) - FL - FCL(256,10) - SM - CE	0.03	75	[-0.067,0.067]	N	N	56.8%	42.6%
MODEL 9 CL(5) - MPL(5,2) - FL - FCL(256,10) - Sig - LL	0.05	75	[-0.067,0.067]	N	N	48.05%	36.3%
MODEL 10 CL(3) - MPL(5,2), CL(5) - MPL(3,2) - FL - FCL(49,10) - SM - CE	0.05	75	[-0.067,0.067]	N	N	25.22%	20.34%
MODEL 11 CL(5) - MPL(3,2) - FL - FCL(289,75) - ReLu - FCL(75,10) - SM - CE	0.003	100	[-0.067,0.067]	N	Y 30/batch	77.95%	44.64%

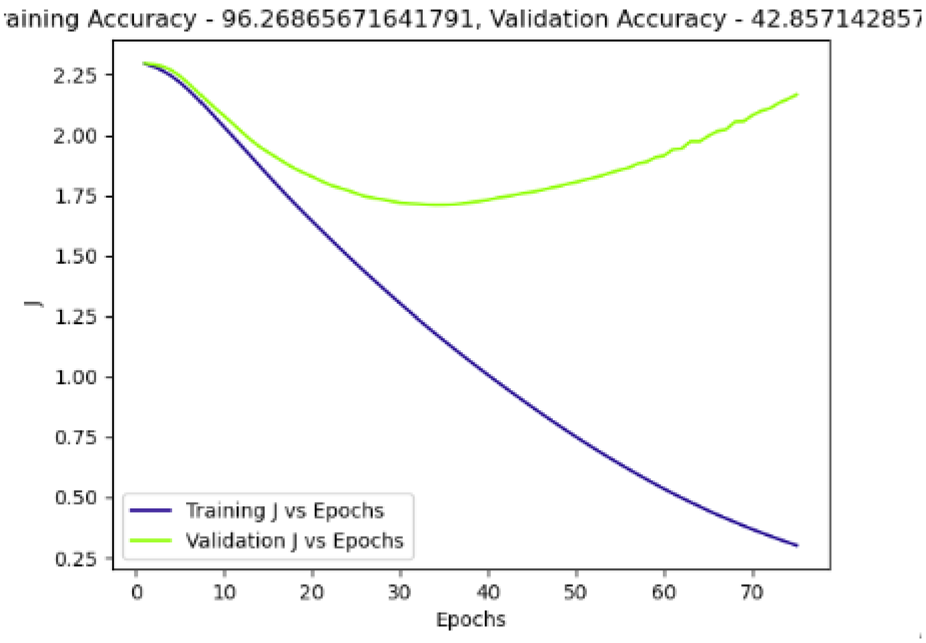

Fig. 1. Model 2

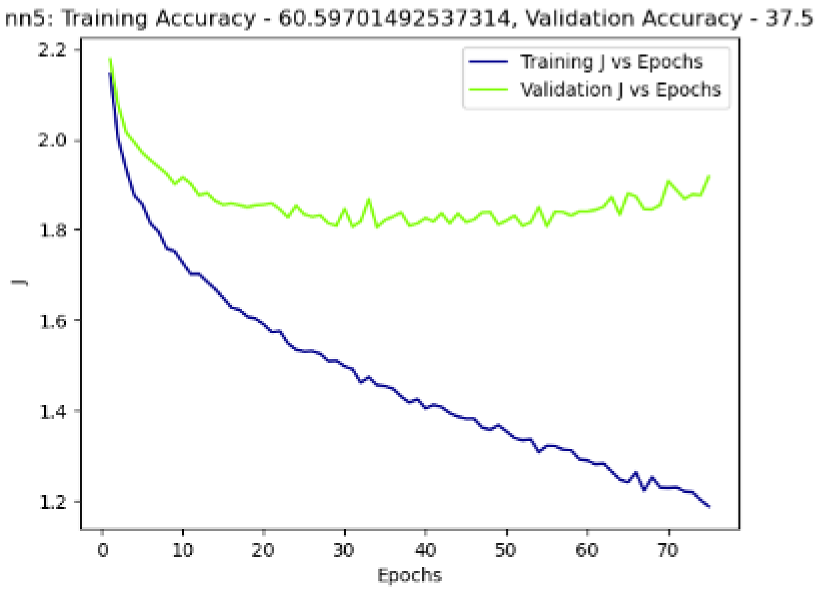

Fig. 2. Model 5

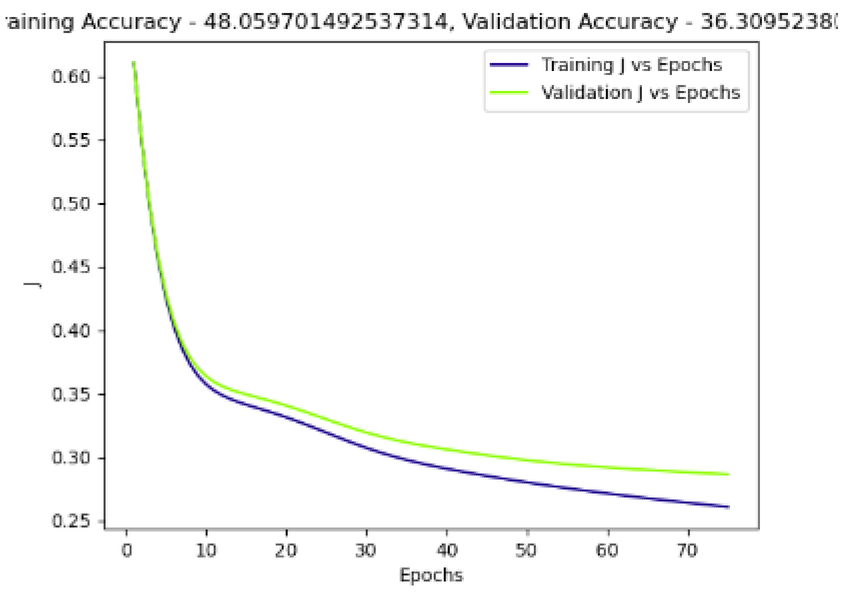

Fig. 3. Model 9

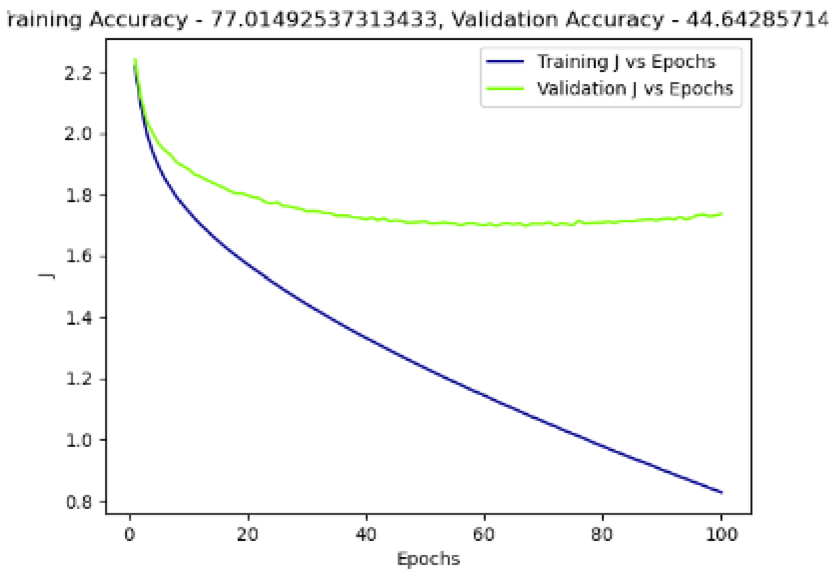

Fig. 4. Model 11

VII. Analysis and Conclusions

We started our experiments with a simple CNN architecture and found that the model was significantly overfitting. We checked to see if any misclassification could have caused this issue but didn't find that to be the case. Later we altered the learning rate, FC weight initialisations along with the size of kernel window for our convolutional layer but the results did not improve significantly. To solve the overfitting problem, we implemented a dropout layer to facilitate regularization but didn't see overall improvements in the accuracy. Rather, we found that models that had dropout layers were performing comparatively bad on training sets for comparable models without the dropout layers.

VIII. Future Work

Based on our observations, the model is learning and the results are above random results but there can be a way to improve the model. We suspect less number of observations per class (about 100 images) and less number of overall observations (about 1000 images) might be responsible. We conclude that to overcome issues like overfitting, we might have to explore various ways like data augmentations and cross validation and data cleaning to improve the results given the scarcity of data on an overall and per class basis.

The reason can also be that the data used for training when resized contains noise and doesn't remain high resolution. Therefore we can also work on high resolution small size images by pre-processing more in the future while incorporating cross validation or creating various "synthetic" observations created by slight variations of ones already in the training set to augment our data and help the learning process.

References

[1] T. Grüter, M. Grüter *et al.*, "Neural and genetic foundations of face recognition and prosopagnosia," *Journal of Neuropsychology*, vol. 2, pp. 79–97, 2008. [Online]. Available: 10.1348/174866407X231001; www.bpsjournals.co.uk

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[3] F. Galton, "Personal identification and description," *Nature*, pp. 173–177, 1888.

[4] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech and time-series," *The handbook of brain theory and neural networks*, 1995.

[5] "Face Recognition Based on Lightweight Convolutional Neural Networks," *Information*, vol. 12, no. 5, 2021. [Online]. Available: 10.3390/info12050191

[6] S. Bajpai and G. Mishra, "Real Time Face Recognition with limited training data: Feature Transfer Learning integrating CNN and Sparse Approximation," *bioRxiv*, 2021. [Online]. Available: 10.1101/2021.03.17.435457

[7] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998. [Online]. Available: 10.1109/34.655647

[8] A. Tolba, A. El-Baz, and A. El-Harby, "Face Recognition: A Literature Review," *International Journal of Signal Processing*, vol. 2, pp. 88–103, 01 2005.

[9] R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: a survey," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705–741, 1995. [Online]. Available: 10.1109/5.381842

[10] A. Mohamed, Y. Weng, J. Jiang, and S. Ipson, "Face detection based neural networks using robust skin color segmentation," in *2008 5th International Multi-Conference on Systems, Signals and Devices*, 2008, pp. 1–5. [Online]. Available: 10.1109/SSD.2008.4632827