Parameter Tuning and Architecture Design for Face Recognition Using Convolutional Neural Networks

Nil Beserler
University of California San Diego
nbeserle@ucsd.edu

Abstract

In recent years deep learning, especially Convolutional Neural Networks (CNNs) has shown exponential progress in face recognition tasks and achieved state-of-the-art results. Face recognition involves identifying an individual given their pictures/videos. It is widely used in many industries, from law enforcement to social media to entertainment. This task can be accomplished by machine learning models such as CNNs, Transfer Learning Models, or Recurrent Neural Networks. This paper proposes two CNN architectures and evaluates their performance on the Labeled Faces in the Wild dataset, additionally, parameter-efficient fine-tuning and various techniques will be used to improve the performance of the models and make them more computationally efficient and accurate. Labeled Faces in the Wild is a public benchmark dataset for such a task it contains over 13,000 images of labeled faces.

1 Introduction

Convolutional Neural Networks are deep networks that are particularly useful for dealing with structured data including images, videos, and text. It's efficiency in such data stems from its convolutional layers that extract features from the input data via its kernel weights that are trained during the training stage, followed by pooling layers that reduce dimensions, an activation function that introduces non-linearity to the network and fully connected layers that performs classification/ regression to the flattened output. Additionally, it can include dropout layers and batch normalization layers that can improve and speed up the performance of models.In this paper CNN's will be used for face recognition across the Labeled Faces in the Wild Dataset. Despite the promising performance achieved by CNNs, it remains unclear how to design a 'good' CNN architecture for a specific classification task due to the lack of theoretical guidance (Lawrence). The methods used in the paper will attempt to adress this issue for the specific dataset.

2 Methods

This paper examines methods to achieve state of art results by fine-tuning the parameters; optimizer, learning rate, and batch size; and discussing architecture design; a 3-layer convolutional neural network and ResNet-18 architecture using the Labeled Faces in The Wild Dataset.

3 Dataset description

The dataset used in this paper is Labeled Faces in the Wild. Labeled Faces in the Wild is a dataset prepared by the University of Massachusetts, Amherst for unconstrained face recognition studies (Labeled). It includes 13,233 images of 5,749 people, and there are 1680 people with two or more images. Images are acquired from the web and are deep funneled to reduce variability. The dataset is considered a benchmark dataset meaning it is compiled for model training/testing purposes.

To have better training results and for computational efficiency, the dataset has been filtered; there are at least 20 samples per person. The images are resized to 150x150 pixels and normalized. 80% of the data is used for training and the remaining 20% is used for testing.

Figure 1: Target labels and three images from the dataset

4 Initial Architecture

The initial architecture trained for the task is as follows; 3 convolutional layers each followed by a max-pooling layer with a 2x2 kernel with a stride size of 2. The output from the last pooling layer is flattened and passed through a fully connected layer, a dropout layer with a rate of 0.5, and another fully connected layer with 62 neurons, one neuron for each class. The fully connected layers also contain a ReLu activation function that introduces non-linearity.

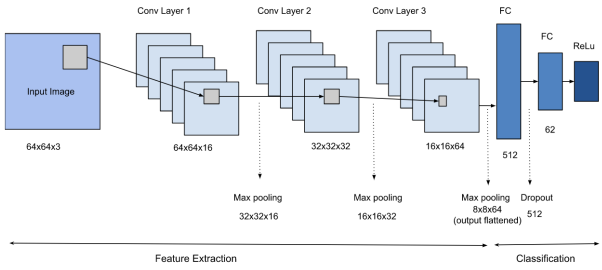

Figure 2: The architecture of the implemented CNN

This convolutional neural network is trained using a Stochastic Gradient Descent optimizer with a momentum of 0.9, a learning rate of 0.001, and Cross Entropy Loss. Some of the predictions for the images can be visualized as:

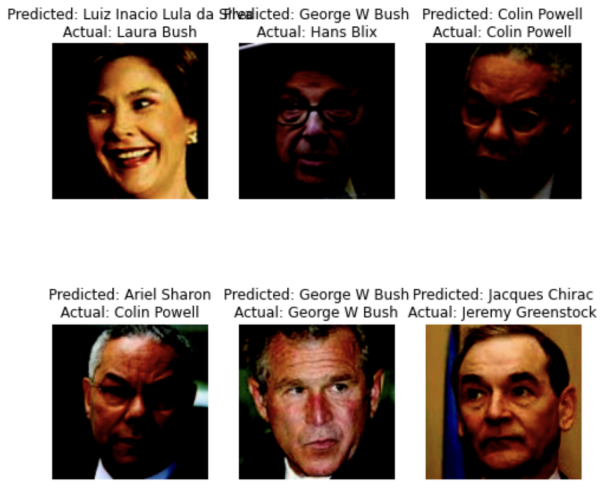

Figure 3: Target labels and predictions using the 3-layer CNN

The accuracy of the model on the test images is 33% which can be considered reasonable given the task difficulty. There exists a trend downward in the mini-batch losses meaning the loss decreases overtime, however, isn't smooth and the convergence is not observable. This suggests that there is great room for improvement.

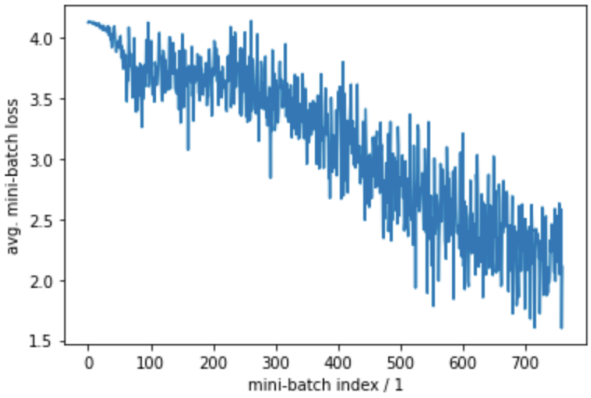

Figure 4: Loss Curve for 3-Layer CNN

5 Fine-tuning Parameters

5.1 Optimizers

Optimizers play a critical role in performance since it's responsible for "learning" by updating weights across the model and minimizing the loss function.

5.1.1 Stochastic Gradient Descent(SGD)

The initial model uses stochastic gradient descent, this is a very common optimizer for Convolutional Neural Networks. The learning rate of the optimizer is 0.001 and the momentum is 0.9. The stochastic gradient descent uses mini-batches of the dataset to calculate gradients of the loss function with respect to weights and update the model weights. With this model, the model has an accuracy score of 33%.

5.1.2 Adaptive Moment Estimation (ADAM):

ADAM is another common optimizer for Convolutional Neural Networks, it is essentially and extended stochastic gradient descent optimizer, it differs from stochastic gradient descent by using the first and second moment of the gradient and an adaptive learning rate to the gradients.There is no single learning rate used across all parameters. The CNN architecture explained previously in combination with ADAM has a model accuracy of 61% which can be considered a great improvement.

5.1.3 AdaDelta

Similar to ADAM, Adadelta also uses adaptive learning rates for each parameter. This optimizer uses root mean squares instead of first and second moments to update the gradients. Momentum is also integrated into Adadelta to accelerate the convergence. AdaDelta has an accuracy score of 16%.

5.1.4 RMSprop

Developed as an improvement to the AdaGrad, Root Mean Square Propagation works by making adjustments to each learning rate for the weights based on the squares of the gradients maintaining a moving average. Implementing this optimizer the model has an accuracy score of 48%.

There are various reasons ADAM optimizer might be performing better than the other optimizers in this particular dataset and model. Some of these possible reasons include:

- Adaptive Learning Rates: As mentioned above ADAM uses first and second-order moments, adjusting the learning rate for each parameter one by one.
- Momentum: The momentum helps the model to converge faster to an optimal solution and avoid getting stuck at local minimums.
- Bias Correction: ADAM includes a bias correction step for both of the moments, to account for initialization. Insofar, RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances. [...] its bias-correction helps Adam slightly outperform RMSprop towards the end of optimization as gradients become sparser. Insofar, Adam might be the best overall choice(Ruder).
- Computational efficiency: Compared to AdaDelta and RMS requires less computational resources, which can lead to faster runtime and computational efficiency.
- Robustness: Adam uses second moment to normalize gradients, and this helps the process of reducing noise and avoiding overfitting.

5.2 Learning rate

Different learning rates can be used for CNN's and there isn't an optimal learning rate that fits all models, the learning rates have to be tested and fine-tuned in order to find the most accurate learning rate. When the learning rate is too low, the model converges slowly, and for the high learning rate, the model training diverges resulting in suboptimal solutions(Johny). This is due to the fact that learning rate determines step size to descend the gradient. Too big of a step size might result in no convergence since it's updated too much at a time or if the step size is too small it might take long periods of time to optimize the model or the model might even get stuck in a local minimum. Testing various learning rates the most accurate learning rate was found to be 0.01 for the given model using Stochastic Gradient Descent optimizer .

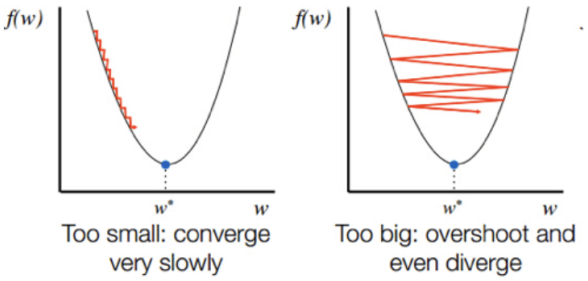

Figure 5: Difference between small and large learning rate (Gomez,2020)

Learning Rate:	0.0001	0.001	0.01	0.1	1
Accuracy Score:	16%	34%	**55%**	16%	1%

5.3 Batch size

The batch size determines the number of samples trained at each iteration. Some common batch sizes are 16, 32,64, and 128 as tested below. The batch size needs to have a balance between computation efficiency and accuracy; larger batch sizes require more computational resources but are more accurate whereas smaller batch sizes require less computational resources but it's less accurate, it is preferred for smaller datasets. There are other factors in play too; very small batch sizes can lead to no convergence, while very big ones can be not generalizable to new data meaning they are overfitting. Therefore this section tests out different batch sizes to determine the optimal one for this model.

Learning Rate:	16	32	64	128
Accuracy Score:	40%	34%	16%	16%

Specific to this model 16 is the best-performing batch size, this could be due to the size of the dataset, and the limited computational resources.

6 ResNet-18 Architecture

Selecting and designing the architecture of the model is also a significant step to solve image recognition problems. In this section a new architecture, ResNet-18 will be introduced and trained on the data. ResNet introduced by He et al. in 2015 is a neural network architecture that has state-of-the-art results on datasets like ImageNet, CIFAR-10, and CIFAR100.

In this specific case, the model is pre-trained on Imagenet and imported from torch-vision models. ResNet-18 consists of 18 layers, 16 of which are convolutional layers and 2 fully connected layers. Additionally, it contains residual blocks which are unique to this architecture. Residual blocks help the model to learn residual mappings between input and output and with the addition of the identity function, a final output can be generated. The vanishing gradient problem, which happens due to the gradients having really small values in backpropagation, is also addressed with the use of ResNet-18 and residual blocks.

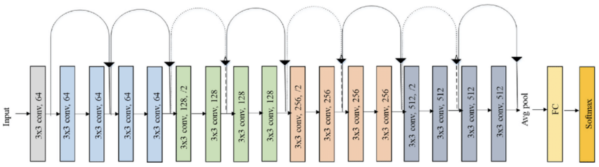

Figure 6: Visualization of ResNet-18 Architecture (Ramzan, 2019)

Training this model with the same hyperparameters as the initial architecture; Stochastic Gradient Descent with learning rate of 0.001,momentum of 0.9, and Cross Entropy Loss results in 88% accuracy with a smooth convergence:

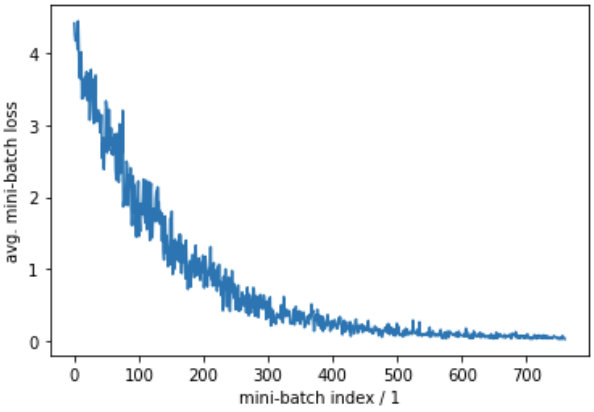

Figure 7: Loss Curve for ResNet-18

The improved performance of RESENT-18 is due to:

- Depth of the architecture: Resnet has 18 convolutional layers while the initial CNN had 3 layers; the depth of a CNN determines the ability to learn complex patterns. However, too many layers can lead to a vanishing gradients problem, as discussed before ResNet addresses this issue with residual blocks. Having more layers also has other disadvantages such as the possibility of overfitting and the need for computational resources. But in this specific dataset; ResNet-18 performs very well by addressing such issues in various ways such as batch normalization and residual connections.

- The number of channels: The previous CNN uses fixed filters while ResNet-18 increases the number of channels from 64 to 512. The more channels the architecture has it can extract more complex features. Again in this case it is important to have an optimal number of channels to avoid problems such as overfitting.

- Residual connections: The gradient is calculated more efficiently and flows better due to the residual blocks.

  The predictions and true labels for six examples can be visualized as:

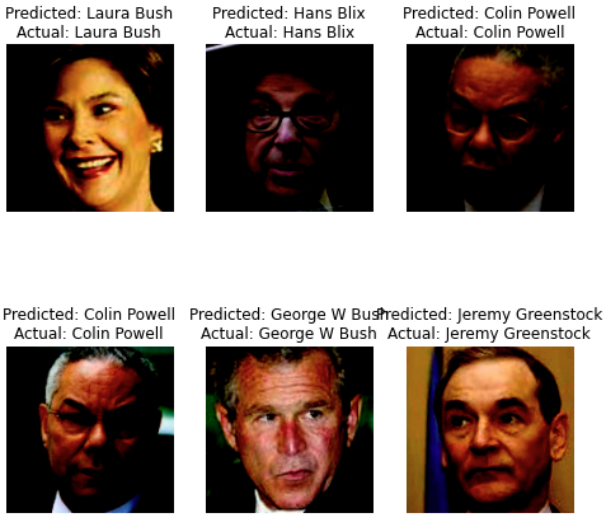

Figure 8: Target labels and predictions using the ResNet-18

7 Results and Conclusion

Both CNN models were successfully trained and were able to classify the images, through out the paper different optimizers, learning rates, batch sizes, and architectures were tested. The 3-layer CNN performed the best using the Adam optimizer, with a learning rate of 0.01, and batch size of 16. Whereas the ResNet-18 architecture performed better overall reaching an accuracy score of 88%. Fine-tuning parameters and architecture design were found to significantly affect the performance of the models.

8 Future Directions

Grid Search can be done across the parameters of the CNN, this could be computationally expensive but would give more optimal parameters for the model that could be used in combination. Other parameters such as; activation functions, number of layers, and dropout layers could be fine-tuned. In addition, other loss functions could be tested with the model. Architectures such as FaceNet, VGG, or ArcFace could be used to get better accuracy scores, and state-of-the-art results, however, their interpretability could potentially be more challenging.

9 Acknowledgments and References

9.1 Acknowledgments

ResNet-18 model was implemented via torchvision's pretrained model: https://pytorch.org/vision/main/models /generated/torchvision.models.resnet18.html

9.2 References

1. Gómez Blas, N., de Mingo López, L. F., Arteta Albert, A., Martínez Llamas, J. (2020). Image classification with convolutional neural networks using Gulf of Maine Humpback Whale Catalog. *Electronics, 9*(5), 731. https://doi.org/10.3390/electronics9050731

2. Johny, A., & Madhusoodanan, K. N. (2021). Dynamic learning rate in deep CNN model for metastasis detection and classification of histopathology images. *Computational and Mathematical Methods in Medicine, 2021*, 1–13. https://doi.org/10.1155/2021/5557168

3. Lawrence, S., Giles, C. L., Ah Chung Tsoi, & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks, 8*(1), 98–113. https://doi.org/10.1109/72.554195

4. Labeled faces in the wild home. LFW Face Database: Main. (n.d.). Retrieved March 18, 2023, from http://vis-www.cs.umass.edu/lfw/

5. Ramzan, F., Khan, M. U., Rehmat, A., Iqbal, S., Saba, T., Rehman, A., & Mehmood, Z. (2019). A deep learning approach for automated diagnosis and Multi-class classification of Alzheimer's disease stages using resting-state fMRI and residual neural networks. *Journal of Medical Systems, 44*(2). https://doi.org/10.1007/s10916-019-1475-2

6. Ruder, S. (2017). An overview of gradient descent optimization algorithm. Cornell University ArXiv. endenumerate endhangindent