Simple Convolution Neural Network for Multi-class Image Classification

Dhruvil Naik
M.Sc Data Science
Loughborough University
Loughborough, United Kingdom

***Abstract*—Nature has inspired countless inventions, birds have inspired us to fly and fish have inspired us to swim. It was with this logic the concept of Artificial Neural Network (ANN) was developed which through research developed in study of Deep Neural Networks inspired by human visual cortex system. In this article we prepared a basic architecture of Convolutional Neural Network for a Multi-class image classification and fine tune the parameters such as learning rate to find the most optimal parameters for accurate class predictions. The baseline model did not achieve a great validation score signifying overfitting. Using Normalization and Regularization techniques did not show any significant improvement in model accuracy. However, applying transfer learning methods using VGG-16 architecture improved the validation accuracy by 32%.**

***Index Terms*—Deep Learning, CNN, Image classification, Transfer Learning**

I. Introduction

Machine Learning and Artificial Intelligence have gained popularity in the recent years with development of image processing, segmentation, feature extraction and object identification [1]. Image classification is one such field that has been a especially popular research topic in recent years. It is the key concept in the area of computer vision . The applications of it have aided scientific experiments, medical imaging equipment and manufacturing of self-driving cars [1].

Artificial Neural Networks (ANN) were initially inspired from network of biological neurons present in the human brain. ANNs are at the very core of deep learning and were introduced in 1943 by a neuropysiologist, Warren McCulloch and Walter Pitts in their paper [4]. The spark in research interest led to introduction of Convolutional Neural Networks (CNN) which is widely known for its pattern recognition and image classification capabilities [5].

CNN architecture is one of the most popular Deep learning framework, it was first proposed in the 1990s. However, due to the lack of computational power and rise of Support Vector Machines(SVM) in the early 200s led to a decline in usage of it. The resurgence of CNN's took place in 2012 when Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition with a deep convolutional neural network(D-CNN) that trained 1.2 million labelled images [6]. This breakthrough demonstrated the power of deep learning and sparked a renewed interest in CNN's.

This paper introduces a basic convolutional neural network (CNN) for an image classification task and examining the impact of tuning hyperparameters on model performance. Specifically, analyzing effects of different methods of learning rate settings and optimization algorithms in quest of optimal parameters and reduce overfitting on training set. It also explores the effect of compositions of different CNN architectures on the outcome of image classification.

II. Data and Preliminary Analysis

The classification task explores 9469 images samples belonging to 10 classes The purpose of this exploratory analysis is to develop an understanding for the underlying structure of the dataset and feature extraction.

The dataset contain 9469 images in RGB color scale belonging to 10 classes, each with a resolution of 500 x 375 pixel. The training set has 9469 samples, each labelled with one of the ten classes: ['Fish', 'Dog', 'Radio', 'Chainsaw', 'Monument', 'Truck', 'Trumpet', 'Golfball', 'Parachute']. The classes are distributed equally around 10% images per class.

It was observed that images are highly varied in terms of content, as some images contained multiple objects e.g. a human holding a fish. The size of the images varied slightly and the size of the target object appearing too far from the camera. It was also observed that the class labels were coded, to overcome this the class folders were relabelled to make object identification easier.

One potential limitation of this dataset is that the sample, although being small for DNN modelling, may seem too large due to the limited computational power available. To overcome this, the training set was split further using 80/20 cross-validation technique and the validation set was kept for evaluating model performance

III. Convolution Neural Networks

A simple CNN model consists mainly of three types of layers, Convolutional, Pooling and Fully-connected. The convolution layer applies a set of trainable filters to the input image, to extract features relevant to the given task. The pooling layers reduces the spatial dimensions of the feature maps generated, whereas activation layer applies a non-linear function to the output of previous layers. Each of these layers will be discussed in further detail below

A. *Overall Architecture*

The CNN does image recognition by breaking down the image into smaller sub-components called receptive fields and analyze each of them individually to extract features. The features are then combined to interpret the image [7].

Convolution layer is at the core layer of CNN. In this layer a set of filter is applied to the input image for feature extraction. Each filter is essentially a weight matrix which slides over the image pixels and generates a resultant matrix with the dot product of the two matrices, the resultant matrix is called a feature map [8].

Initialization strategy was introduced in a 2010 paper by Xavier Glorot and Yoshua Bengion [9], to solve a problem called vanishing/ exploding gradients. CNN's work on a principle of back-propagation, a technique in which the error is propagated back through the output propagating the error gradient along the way. Unfortunately in dense models the gradients often smaller gradually and leaves the lower layers unchanged. To overcome this Glorot and Bengion suggested that making the variance equal for both input and output layers this problem can be solved. Table I below explains the strategy which can be applied at input layer to overcome the vanishing/exploding gradient problem. The classification model in this paper makes use of the `he_uniform` initializer along with the `ReLU` (Rectified Linear Unit) activation function for faster convergence [11].

TABLE I
INITIALIZATION PARAMETERS FOR DIFFERENT ACTIVATIONS

Initializer	Activations	σ^2(Normal)
He	None, tanh, sigmoid, softmax	$2/fan_{in}$
Glorot	ReLU, Leaky ReLU	$1/fan_{avg}$

Non-Linear activation functions are used at every node of the CNN layers to accept the output values from previous layers neurons. A non-linear activation introduces non-linearity into the model to improve the accuracy and fit the model better. In the given classification task `ReLU` was used as the activation function. Mathematically it is defined as $f(x) = max(0, x)$. It identifies negative values and returns them as zero and remains linear for all positive value. It is popularly used in DNN applications due to its simplicity which requires less computational power.

Pooling is the next step in the convolution process, same computation occurs by taking average (Average Pooling) or maximum (Max Pooling) of the two matrices, this is done for preservation of features and increase the robustness of feature extraction [7] [1]. High level features can be extracted from the input layers by stacking convolutional and pooling layers [1].

Finally, in a **fully-connected layer** the reduced sub-images are combined to perform classification. A softmax activation function classifies a multiple classes by assigning a decimal probability to each class. The sum of probabilities assigned to each class should sum to 1.

IV. METHODS, EXPERIMENTS AND ANALYSIS

A. *Baseline model*

The Baseline model is a reliably simple architecture which consists of 6 layers of 2 convolution, pooling and fully connected each. It uses a standard 32 filters with a kernel size of 3x3. The Pooling layer searches for the maximum value and slides the kernel with a stride of 2 to reduce the number of feature maps for faster computation. Fully connected layer has 128 dense connections which narrows to 10 output neurons which is the number of classes.

It is clear from a first glance at Fig.1 that the model performs extremely well on training data but achieves 58.34% accuracy on validation set. The CNN model overfits the training data and will need further optimization. Further optimization techniques applied will be discussed in the later sections.

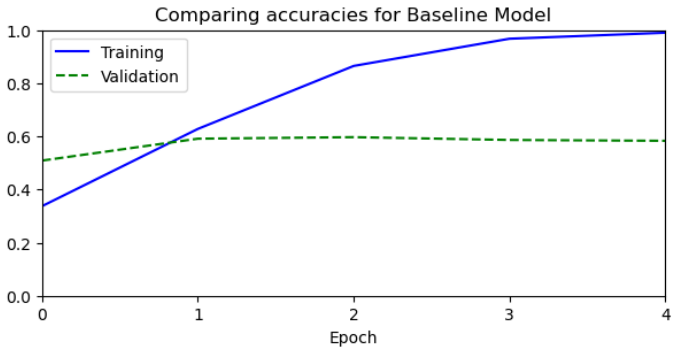

Fig. 1. Baseline Model Performance

B. *Faster Optimizers*

Training a Deep neural network can be a challenging task. In the recent years, there has been a rise in exploring faster optimization techniques, this section will compare three different optimizers tried on the given image classification task and compare the results.

*1) **Adaptive Moment Estimation (Adam)**:* Adaptive moment estimation, combines the ideas of momentum optimization. Adam is a computationally effective algorithm that can converge quickly on many deep learning problems. Adaptive Moment Estimation was adapted from RMSProp and SGD with momentum. Adam updates the learning rate for each network weight individually [10] [11]. Unlike RMSProp and Adadelta, Adam saves both the exponentially decaying average of past gradients as well(m_t) as past squared gradients (v_t) [10].

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

In the above equations $\beta 1$ and $\beta 2$ represent the decay rates of the moving averages and m_t, v_t are first and second moment of gradients [10]

Adam is used extensively due to being computationally efficient and straightforward implementation, Adam can achieve

good model performance without the need for hyperparameter tuning.

*2) **RMSProp**:* Root Mean Square Propogation is another optimizer works on a principle of reducing the number of evaluations to reach local optimum. RMSProp aims to penalize the update of the parameter that causes the cost function to oscillate a lot

$v(w,t) := \gamma(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2$

*3) **Stochastic Gradient Descent (SGD)**:* Stochastic Gradient Descent is an extension of Gradient Descent algorithm which comes with the ability to work with non-convex optimization problems. SGD algorithm performs one update at a time of entire sample rather than batch processing, leading to faster convergence. However, it requires a perfect balance of learning rate as it could lead to slower convergence for a smaller learning rate or overshoot the minima if the learning rate is too high [11].

$$\theta = \theta - \alpha \frac{\partial}{\partial \theta} J(\theta; x^{(i)}, y^{(i)})$$

The gradient update step in SGD is performed using the above equation where $x^{(i)}$ and $y^{(i)}$ are training examples.

*4) **Comparing Optimizers**:* We trained three different models, with additional layers on top of Baseline model and ran them for 5 epochs each. The model architecture consists of 10 layers in total which include 2 convolution layers and 2 fully connected layers as well as Batch Normalization and Dropout Layers. The initial plan to use GridSearch methods was unsuccessful due to insufficient computational power. it is evident from Fig.2 that all the models performed equally well on both training and validation sets. However, `RMSProp` achieved the highest accuracy of 98.7% amongst the three models. The `RMSProp` optimizer converges faster with minimal loss compared to the other optimizers. `RMSProp` model will be used to fine tune the model further using different methods which will be discussed in the later sections.

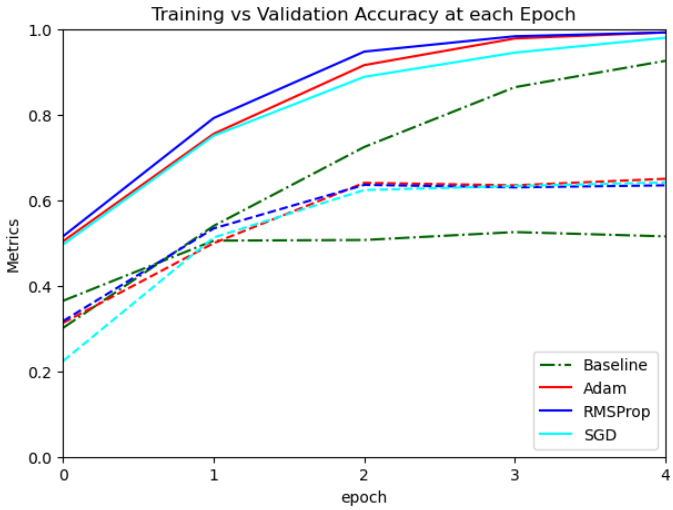

Fig. 2. comparing model performance of different optimizers

C. Batch Normalization

For very deep neural networks, minor changes in the previous layers build up and increase the risk of vanishing/exploding gradients. The change in distribution of layers inputs can cause slow convergence and poor generalization because the network parameters have to adapt to new distribution with each iteration, this is know as the Internal Covariate Shift [6]. Sergey Ioffe, in a 2015 paper, proposed a new technique called *batch normalization* (BN) that addresses there problems [2]. He suggested the that every hidden layers inputs be normalized and then scaled and shift the results using two new parameter vectors per layers: one for scaling and one for shifting [2]

$$\hat{x_k} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

$$z^{(i)} = \gamma \hat{x}^{(i)} + \beta$$

In the above formula, $\hat{x}^{(i)}$ is the vector of normalized inputs for instance i, ε is a smoothing term to prevent exploding gradients.

Results from the table below show that Batch Normalization has deteriorated the convergence speed as well as decrease the model performance on validation set. Although, Training data shows promising results the model tends to overfit as the regularization techniques have not been applied to the model.

TABLE II
COMPARING RMSPROP PERFORMANCE

Epoch		1	5	10
val_loss	RMSProp	2.226	1.639	-
	RMSProp_BN	3.565	1.687	2.03
val_accuracy	RMSProp	0.317	0.634	-
	RMSProp_BN	0.161	0.589	0.604
Time	RMSProp	723s	246s	220s
	RMSProp_BN	288s	202s	220s

D. Dropout

Dropout is one of the most popular regularization technique in Deep neural networks. The idea behind it that every neuron including the input neurons will be randomly assigned a probability p of being "dropped-out"/switched to zero, essentially forcing the remaining neurons to learn stronger features. The network is able to identify noisy pixels in the data, as it less sensitive to specific weights of individual data. The hyperparameter p is called the dropout rate and is typically set between 40%-50% in Convolutional Neural Networks [2].

Dropout method is so powerful that it can generate a unique network at each training step. A single neuron can be in two possible states which creates a combination 2^n possible networks. However, only a handful of combinations are explored during training [2].

The given classification task, we set up a Dropout layer at each convolution layer with a varying probability ranging from 0.3-0.5% alongwith Batch Normalization(BN). A sharp drop in model performance was seen. We noted a drop in

validation accuracy from 60.19% in the original model to 56.55% in the optimized model. The drop in performance could be attributed to the strong dropout setting explored. A softer dropout rate could possibly lead to improved accuracy and faster convergence with Batch Normalization(BN) which could not be explored in this current round of study performed.

E. Transfer Learning (VGG16 Architecture)

Machine learning and Deep learning come with numerous set of challenges, these algorithms have conventionally been designed to work for a specific feature space distribution which means models need to rebuilt and evaluated from scratch. Additionally, DNN's require labelled data for better performance which is often expensive, time-consuming or even unrealistic [13]. Transfer learning by allowing knowledge transfer across different domain possessing similar characteristics would significantly improve the performance of learning [12].

Fig.3 below shows comparison between the baseline RMSProp model and a pretrained model based on the VGG-16 architecture. The VGG-16 outperforms the baseline model by a large margin achieving close to 90% accuracy on validation set which indicates that the VGG-16 did not overfit to the validation data. The pretrained model was able to learn more complex features and generalize better to the validation set. However, model performance should be further evaluated on test set to comfirm accuracies.

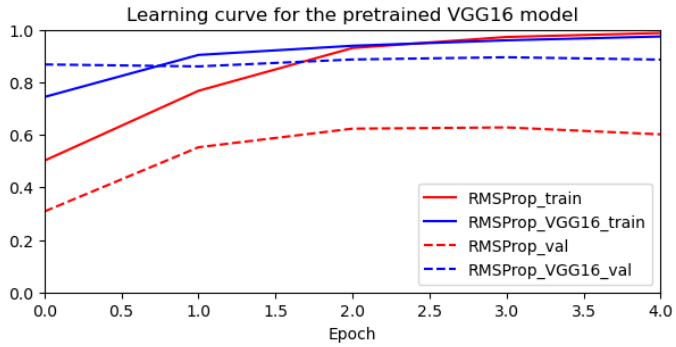

Fig. 3. Comparing Pretrained vs Baseline Model

V. Conclusion

In this article, we discovered the architecture of a Convolutional Neural Network and how deep learning models learn features from input images to be able to correctly classify input images. We began by building a simple CNN architecture and explored various hyperparrameter tuning techniques beginning with Batch Normalization(BN) for faster convergence. Next, we explored three optimization techniques to improve the speed and accuracy of a deep learning model by minimizing the loss function. We further explored regularization technique to prevent overfitting the data and eventually explored a new concept of transfer learning which significantly improves model performance by applying the optimal set of parameters achieved from a model in a similar domain and characteristics.

Due to the limited computation power, we were not able to explore the best set of parameters for this current model. However, the key finding was that transfer learning model improved model performance significantly whilst using less memory.

Apart from the techniques we discussed, deep learning models can be optimized using many other techniques which will require further work.

References

[1] T. Guo, J. Dong, H. Li and Y. Gao, "Simple convolutional neural network on image classification," 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, China, 2017, pp. 721-724, doi: 10.1109/ICBDA.2017.8078730.

[2] Geron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (3rd ed.). O'Reilly.

[3] Alake, R. (2022, March 5). Understanding and implementing lenet-5 CNN Architecture (Deep Learning). Medium. Retrieved March 15, 2023, from https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342

[4] McCulloch, W., & Pitts, W. (2021). A logical calculus of the ideas immanent in nervous activity (1943). Ideas That Created the Future, 79–88. https://doi.org/10.7551/mitpress/12274.003.0011

[5] O'Shea, K., amp; Nash, R. (2015, December 2). An introduction to Convolutional Neural Networks. arXiv.org. Retrieved March 15, 2023, from https://arxiv.org/abs/1511.08458

[6] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 2015, pp. 730-734, doi: 10.1109/ACPR.2015.7486599.

[7] Lang, N. (2022, October 24). Using convolutional neural network for Image Classification. Medium. Retrieved March 16, 2023, from https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4

[8] Sharma, D. (2021, January 14). CNN for Image Classification: Image Classification using CNN. Analytics Vidhya. Retrieved March 16, 2023, from https://www.analyticsvidhya.com/blog/2021/01/image-classification-using-convolutional-neural-networks-a-step-by-step-guide/

[9] Glorot, X., amp; Bengio, Y. (2010, January 1). [PDF] understanding the difficulty of training deep feedforward neural networks: Semantic scholar. [PDF] Understanding the difficulty of training deep feedforward neural networks — Semantic Scholar. Retrieved March 16, 2023, from https://www.semanticscholar.org/paper/Understanding-the-difficulty-of-training-deep-Glorot-Bengio/b71ac1e9fb49420d13e084ac67254a0bbd40f83f

[10] Chauhan, N. S. (2020, December 18). Optimization algorithms in Neural Networks. KDnuggets. Retrieved March 17, 2023, from https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html

[11] Gupta, A. (2023, March 3). Optimizers in Deep learning: A comprehensive guide. Analytics Vidhya. Retrieved March 17, 2023, from https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/What_Are_Optimizers_in_Deep_Learning

[12] Tammina, S. (2019). Transfer learning using VGG-16 with deep convolutional neural network for classifying images. International Journal of Scientific and Research Publications (IJSRP), 9(10). https://doi.org/10.29322/ijsrp.9.10.2019.p9420

[13] F. Zhuang et al., "A Comprehensive Survey on Transfer Learning," in Proceedings of the IEEE, vol. 109, no. 1, pp. 43-76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.