A Study of Image Classification using Convolutional Neural Networks

Krishnan Ravichandran
kxr200008@utdallas.edu

Siddharth Sarangi
sxs200077@utdallas.edu

Gokulnath Dayalan
gxd210005@utdallas.edu

***Abstract*—In the last several years, deep neural networks have made significant progress in the field of AI for image classification. The rapid development of deep learning techniques has significantly improved the performance of convolutional neural networks (CNNs) in image classification tasks. CNNs are a type of deep learning model that are commonly used in computer vision tasks for image classification, object detection, and segmentation. These models are designed to learn and extract features from image data and use those features to make predictions or classify images. In this project, we aim to compare the performance of various popular CNN architectures on a standard image classification dataset, and propose new CNN architectures that offer improved accuracy and performance. Additionally, we have carried out a number of exploratory analyses on the dataset used to contrast the models. The models are fairly assessed using apt hyperparameters, and the outcomes are deduced using various evaluation metrics that are tailored to image CNNs.**

***Index Terms*—Image Classification; Convolutional Neural Networks; Deep Learning**

I. Introduction

Neocognitron, the first design of its kind and arguably the earliest precursor of CNNs established the concepts of feature extraction, layer pooling, and using convolution in neural networks before the convolution neural network was created. From there, Yann Lecun initially proposed the Convolution Neural Network, or ConvNet, between 1989 and 1998. Simply said, a convolution neural network is a neural network. Let's now examine convolution in greater detail. To put it simply, convolution is a mathematical technique that enables the fusion of two sets of data. Convolution is used in the context of image classification to extract information from the input image, which serves as the input data and creates a feature map that contains the key data. The feature map, which is also known as a kernel or feature detector, is slid over the image while each element is subjected to matrix multiplication. The 3×3 kernel is typically slid until the feature map is finished. The feature map that is created as a result of the filter operation is used by subsequent layers to find more complex features. When a convolution procedure is used, the image size typically decreases each time. Mathematically speaking, the result is a picture with the size

$$((m-n+1)*(m-n+1))$$

when an m ×m image is convoluted with an n ×n kernel

When a filter operation is used on input data, two issues frequently arise. (i) As per the calculation [ref] , image size decreases. (ii) The original image's corners were less heavily impacted by the kernel than its center. In comparison to the middle, the edges of the image are less frequently employed. . The edges of the image aren't much used as compared to middle. These problems are usually resolved using techniques called padding and striding.

Padding

Padding is used to preserve the original size of the image to overcome the above-mentioned problems by adding layers of zeroes depending on the padding value. If an input data of n×n is convoluted with the filter of a certain size then the result of the operation can be calculated as,

$$(n+2\times p-f+1)\times(n+2\times p-f+1)$$

Striding

Striding is the number of steps that a filter should slid while applying over an input image. CNN usually there involves a lot of layers so more computational power is needed if we use a smaller stride. Furthermore learning high-level features is more important than low-level features in such cases larger stride can be used depending on the dataset and the convolution layers. Generally, the output of padding and striding can be calculated as,

$$\left\lfloor \frac{n+2\times p-f}{s}+1 \right\rfloor \times \left\lfloor \frac{n+2\times p-f}{s}+1 \right\rfloor$$

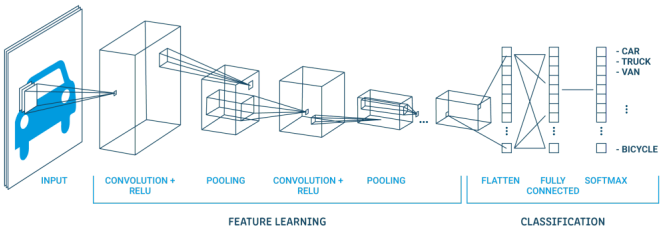

Fig. 1. General CNN Architecture

CNN architecture are usually stacked up of different layers. The layers that build the architecture are discussed as follows,

Convolution Layer

The convolution layer is the operation of a filter over the input image to detect features as discussed in the previous section. Convolution layers in CNN form the core building blocks.

Pooling Layer

This layer's primary goal is to lower the size of the convolved feature map in order to reduce computational expenses. Pooling activities are typically classified into two types: max pooling and average pooling. (I) Max Pooling: It takes the maximum value for each feature map region. (ii) Average Pooling: It computes the average value for each feature map patch.

Fully Connected Layer

The previous layer's flattened output is transferred to the Fully connected layer. Every input unit is connected to every output unit in the FC layer. This layer is often used to interpret the convolutional layers' learnt features and generate a prediction or output for the network.

Activation Function

The activation function classifies the output from the fully connected layer. There is various activation function used in a different type of classification depending upon the dataset or image type used. To put it mathematically, an activation function does a nonlinear transformation on an input data or signal. The output of this function is then passed to subsequent layers for further processing. In our project various binary and multi-class classification activation functions are used. Some of the functions include ReLu, softmax, and sigmoid. There are other activation functions also used according to respective non-linear transformations

II. Related Work

In today's world of computer science, Image Classification is a demanding challenge. Many researchers have tried presenting various deep neural network models to provide a solution for this challenge. The older architecture such as LeNet was designed to provide image classification solution for handwritten digits without the use of GPU and CUDA. Henceforth, it became a legendary model which inspired a lot of research works. AlexNet, with inspiration from LeNet, became widely known for its accuracy measure on the ImageNet dataset. It was also devised for bigger-dimension images. With the introduction of VGGnet, the approach changed a lot. Till here computer vision researchers depended mostly upon increasing the number of trainable parameters, using different activation functions, and more on improvising networks by introducing concepts such as batch-normalization and dropout layers. Then the augment of newer architectures, models such as GoogleNet and ResNet there were a lot of changes to the legacy approach of building image classification models. With inspiration from both the older and the newer architectures, we have tried to provide certain new models and how they perform as compared to these legacy models.

After going through different papers related to this work, we got to know that most of the research people focus on adding new layers. This might be the case because they have large GPU and CPU resources. But we tried focusing more on how can we improvise the legacy models or if we are adding new layers what purpose does it have? The main focus was on introducing more concepts like how batch normalizing, dropout layers, and introducing how with less number of parameters we can develop a model which will take less training time. With this goal, we have presented three new architectures in this paper.

III. Datasets

A. MNIST

MNIST is a large dataset of comparatively easy data. It consists of 28X28 dimension grey-scale images. It contains handwritten digits and was primarily designed with the goal of models which can work on grey-scale images or models which can work on single channels images. MNIST was formed out of re-mixing NIST dataset, which consists of a collection of handwritten digits from various banks as training data and school children's handwritten digits as testing data. MNIST consists of 60,000 training images and 10,000 testing images. It consists of 10 labels, that is, 0 to 9. The mean of the training dataset happens to be 0.1307 and the testing part happens to be 0.3081. Fig 1 represents the content of the dataset.

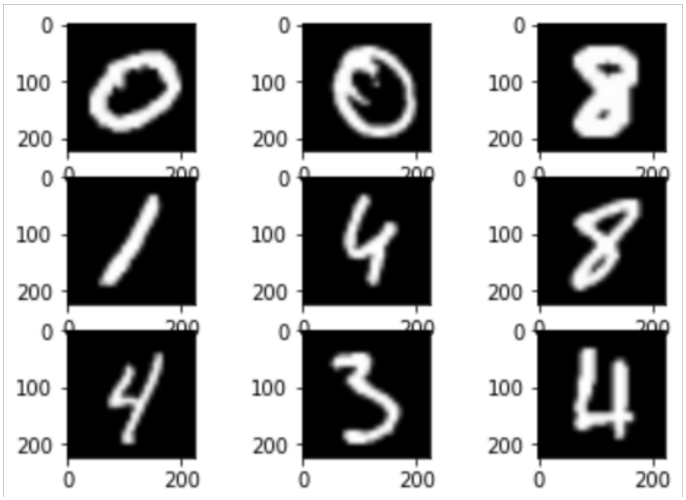

Fig. 2. MNIST sample

B. Fashion-MNIST

Fashion-MNIST was formed to provide a better and complex dataset over MNIST. It was designed by researchers from Zalando Research, with the attitude to provide a benchmark dataset, with the qualities and properties of MNIST dataset. This is very evident from the fact that, the individual imaze size still remains 28X28 and consists of grey-scale images(consisting of single channel). Here also, the training

set consists of 60,000 images and the testing set consists of 10,000 images. But the complexity of the dataset is better as compared to the that of MNIST dataset. The previous statement is proven by applying certain different machine learning modes on both the datasets by the designer of Fashion-MNIST. The following table provides, an average accuracy of various modesl on each of the datasets. This dataset consists of 10 classes. Fig 2 represents the content of the dataset. [6]

Models	MNIST	Fashion-MNIST
RandomForest	0.96	0.85
Preceptron	0.85	0.75
SVC	0.97	0.83

The dataset cleary provides more complexity, increasing the challenge for researchers to understand the features of the images and accordingly to form better models over the present models. The mean and the standard deviations of the dataset is same as that of MNIST.

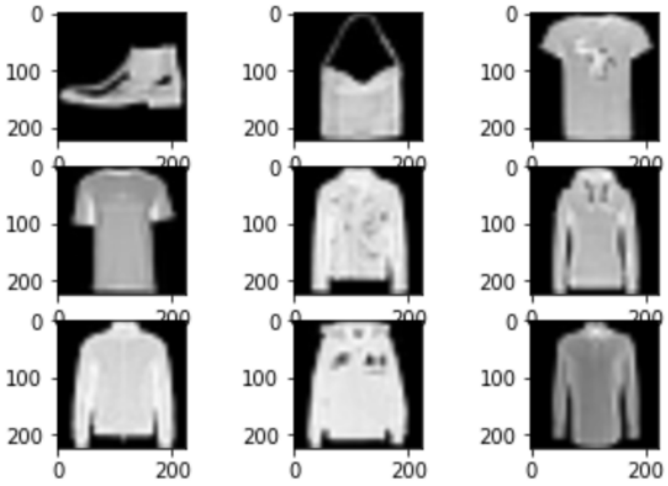

Fig. 3. Fashion-MNIST sample

C. CIFAR-100

CIFAR-100 happens to be the dataset with colorful images. CIFAR stands for Canadian Institute for Advanced Research, and the researchers here collected images to form this dataset. Here the training set consists of 50,000 images and the test set consists of 10,000 images. As images are colorful, the dimension of an individual image is 32X32X3(3 channels). The mean and standard deviation of the training dataset happens to be (125.30691805 122.95039414 113.86538318) and (62.99321928 62.08870764 66.70489964). Also, CIFAR-100 has got 100 classifications, but there is a small catch. Each image comes with two labels, one is a 'fine' label which defines to which class they belong, and the second one is a 'coarse' label which defines to which superclass they belong. The CIFAR-100 dataset provides real-life image complexity and provides an idea of how the designed models work on real-life image classification. Fig 3 represents the content of CIFAR-100.

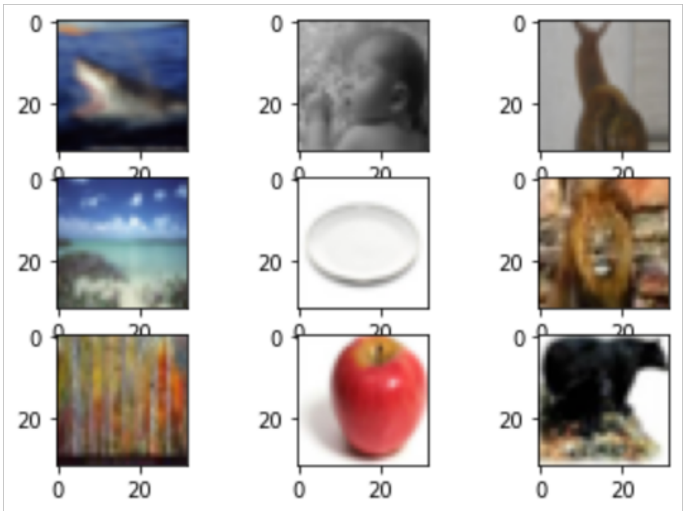

Fig. 4. CIFAR-100 sample

IV. Exploratory Data Analysis

The primary goal of exploratory data analysis is to analyze the dataset by visualizing and summarizing key characteristics such as class distribution, size distribution, and so on. Furthermore, it aids in data compression and the discovery of an informative representation for subsequent processing. Another crucial component of data analysis is that it can aid in dimensionality reduction, also known as feature extraction. Feature extraction is the process of selecting and merging variables into features in order to reduce the quantity of data that must be processed while still thoroughly and accurately characterizing the dataset. The dataset is analyzed using three methods in this project: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-distributed stochastic neighbor embedding (t-SNE) based on supervised or unsupervised learning.

For our dataset, we randomly sampled 10000 images to perform PCA, LDA and t-SNE.

A. LDA

Principle Component Analysis (PCA) and linear discriminant analysis (LDA) are primarily dimensionality reduction techniques. They can alsobe used to effectively visualize data in lower dimensions. The later focuses on the axes that maximize the separation between different classes rather than the component axes that maximize the variance of our data.

The LDA plots of our datasets reveal a clear picture of the complexity of classification. We noticed that the MNIST and FashionMNIST output classes are clustered together. The boundaries in MNIST dataset are very clearly established to a point where simple clustering algorithms can perform very well with these images. The same cannot be said for the CIFAR-100 dataset. The class boundaries are not well defined and we see large overlaps. Even for a human brain, the CIFAR100 images are perplexing due to the fact of its size

being just 32x32. This makes it very blurry for a 3-channel RGB image.

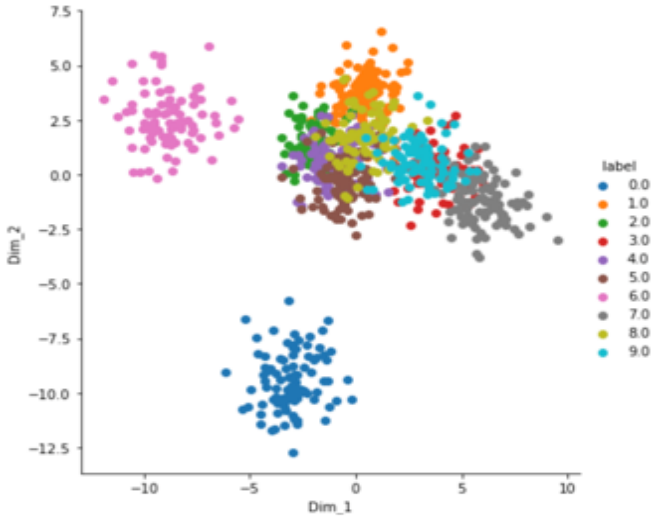

Fig. 5. LDA Visualization of MNIST

B. t-SNE

t-SNE is a nonlinear dimension reduction technique that can handle nonlinear manifold structures. In contrast, PCA is a linear dimension reduction technique that focuses on maintaining large pairwise distances in the projection. The t-SNE algorithm attempts to find a two-dimensional representation of data that best preserves the distances between points The goal of this process is to preserve the local structure of the data so that points that are neighbors in the original feature space will also be neighbors in the two-dimensional representation.

The t-SNE plots reaffirm the conclusion we made from LDA. The features-plot of CIFAR-100 is scattered across and there are no clear visible patterns in the random sample of 10000 we chose. From this analysis, we expect that simple classical models will perform exceedingly well with MNIST, and reasonably well with Fashion-MNIST. Whereas for CIFAR-100 we would require a more complex and deeper set of convolutions supported by other feature-enhancing layers.

V. Training Methodology

The main objective of this project is to compare the performance of different convolution neural networks. Significant measures were taken to control the experimental bias. There was little to no change in the training algorithm, the hyper-parameters and the training environment. On rare occasions, due to environmental constraints, some of the parameters had to be changed to complete the training. One of the major criteria for evaluation was training speed, which was also the motivation behind the choice of many hyper-parameters.

Environment

All the models were trained in Google Colab (Free-tier), with GPU(Cuda) having 12 gigabytes of memory.

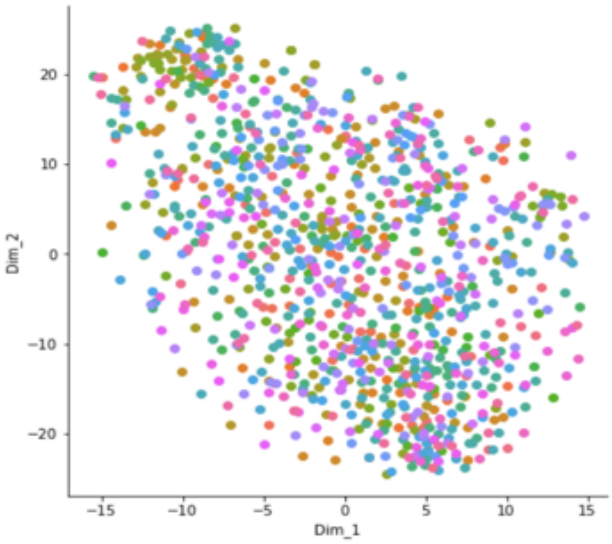

Fig. 6. t-SNE Visualization for CIFAR-100

Hyper-parameters

Some of the important parameters used in the training algorithm is given below

Parameter	Value
Learning rate	0.0001
Optimizer	Adam
Loss function	CrossEntropyLoss
No. of training Epochs	10
Batch size	200

Algorithm and Model selection

The initial training dataset was split into the training (80%) and validation (20%) data. And each epoch had two phases - the training and the validation phase. Loss and accuracy of the predictions were calculated for each phase, with the weights being updated and the end of each training phase. The validation accuracy is recorded for each epoch. The model with the best accuracy across all epochs is chosen as the final trained model. This was done primarily to combat overfitting.

At the end of each training phase, we examine the classification report including the accuracy, F-1 score, recall and confusion matrix. We plot the training and validation accuracies and loss. Considering the length of this report we have included some but not all of the metrics and plots.

It is also worth mentioning that all of the 8 models that we have implemented, terminate with a fully connected layer that generates a verctor of size equal to the number of output classes (10 in the case of MNIST & FashionMNIST and 100 for CIFAR100). And the predicted class is taken as the index with the maximum value. The same methodology is used to get the final predictions of the test set, to calculate the accuracy.

VI. Classical Architectures

A. LeNet

LeNet is one of the earlier architecture in the field of image recognition. It was proposed in the year 1998, and was given by Yann LeCunn. The main motto of the model was to recognition of handwritten characters. Later, it was also improvised to recognise printed number characters too. The best thing of LeNet model was, it came into picture when CUDA was not present in the market.

LeNet consists of five layers. The input is a single channel (grey-scale) 32X32 image. The first two layers are a combination of convolution and average pooling layers, followed by other convolution layers. The activation function used after convolution is Tanh. The final two layers are fully connected layers. From here the distribution for each class is generated. Fig - 7, represents a tabular representation of LeNet. [7]

Layers	Filter Size	Stride	Filters/Neurons	Output Size	Activation Functio
Input				32 X 32 X 1	
Conv - 1	5 X 5	1	6	28 X 28 X 6	Tanh
Avg. Pool -1	2 X 2	2		14 X 14 X 6	
Conv - 2	5 X 5	1	16	10 X 10 X 16	Tanh
Avg. Pool - 2	2 X 2	2		5 X 5 X 16	
Conv - 3	5 X 5	1	120	120	Tanh
FCC - 1	---	---		84	Tanh
FCC - 2	---	---		10	Softmax

Fig. 7. Lenet Architecture

LeNet performs well on the MNIST and Fashion-MNIST, as both the dataset are designed with single channel. And the originally, Lenet was also proposed for grey-scale images. But when CIFAR100 comes into the picture, LeNet is not so competent(Fig - 8).

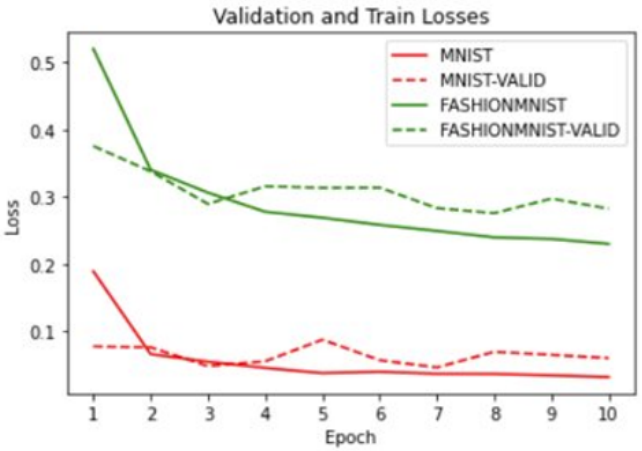

Fig. 8. Training and Validation Loss for MNIST & Fashion-MNIST using LeNet

The model can be improved, if we use Relu function instead of Tanh function for activation function. Also introducing more layers, shows a better accuracy for CIFAR100.

B. AlexNet

In the field of Computer Vision, AlexNet is one of the most appreciated models. It won the Imagenet visual recognition challenge in 2012. Proposed by Alex Krizhevsky and other researchers, it is a very well known image classification model. It was built with more trainable parameters as compared to LeNet, and also the number of layers are more. AlexNet was also designed for input with higher dimensions. AlexNet consists of 63.3 million trainable parameters.

The architecture of AlexNet is very intriguing, as it was the first convolution network with was designed to use GPU for performance boosting. It consists of 5 convolution layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and a softmax layer at the last. AlexNet also introduces dropout layers.

The original model which won the competition was tuned to use Relu as the activation function, with training and validation batch size set to 128. It used stochastic gradient descent with momentum as the training algorithm. The interesting part about the architecture is they used overlapped max-pooling layers. The author states, because they used 3X3 pooling layers with stride 2, they were able to reduce the top-1 error by 0.4% and the top-5 error by 2%. [12]

The introduction to the dropout layer was also very helpful. What Dropout does is, it provides a probability of 0.5 for dropping a neuron from the neural network. This helps the deep neural networks to avoid over-fitting. Also, there was a lot of data augmentation such as mirroring and random crops on the Imagenet dataset. But the main features were the model's ability to use GPU for training a huge number of parameters. AlexNet's model complexity also became the inspiration for VGG-16 and GoogleNets Below is a tabular representation of the AlexNet architecture(Fig - 9).

Layers	Filter Size	Stride	Filters/Neurons	Output Size	Activation Functio
Input				227 X 227 X 3	
Conv - 1	11 X 11	4	96	55 X 55 X 96	Relu
Max Pool -1	3 X 3	2	--	27 X 27 X 96	
Conv - 2	5 X 5	1	256	27 X 27 X 256	Relu
Max Pool -2	3 X 3	2	--	13 X 13 X 256	
Conv - 3	3 X 3	1	384	13 X 13 X 384	Relu
Conv - 4	3 X 3	1	384	13 X 13 X 384	Relu
Conv - 5	3 X 3	1	256	13 X 13 X 256	Relu
Max Pool - 3	3 X 3	2	--	6 X 6 X 256	
Dropout -1	--	--	0.5	6 X 6 X 256	
FCC - 1	--	--	--	4096	Relu
Dropout -2	--	--	0.5	4096	
FCC - 2	--	--	--	4096	Relu
FCC - 3	--	--	--	1000	Softmax

Fig. 9. AlexNet Architecture

AlexNet performs, comparatively better on MNIST and Fashion-MNIST. But it fails to provide a better result for CIFAR100. The following are the training and validation loss for each of the three datasets(Fig - 10).

From the following image, it is very clear that the training and validation loss is not going down as comparatively as it went for MNIST and Fashion-MNIST, may be increasing the number of epochs could have shown a better result, but with respect to the environmental constraints, that we set for each dataset and each model, AlexNet is not so competent on CIFAR-100(Fig - 11).

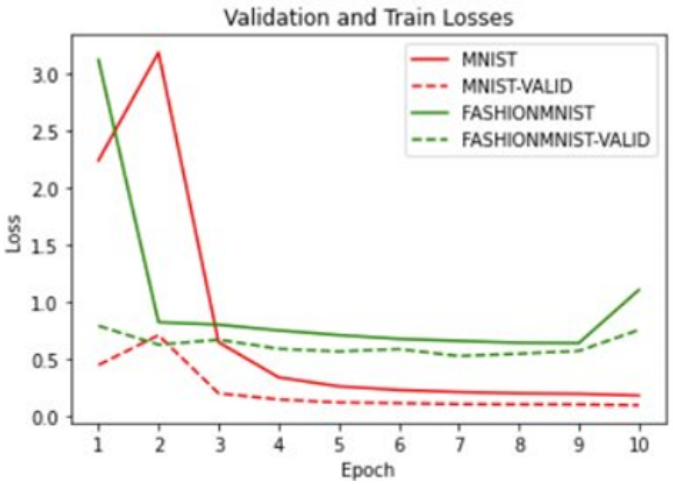

Fig. 10. Training and Validation Loss for MNIST Fashion-MNIST using AlexNet

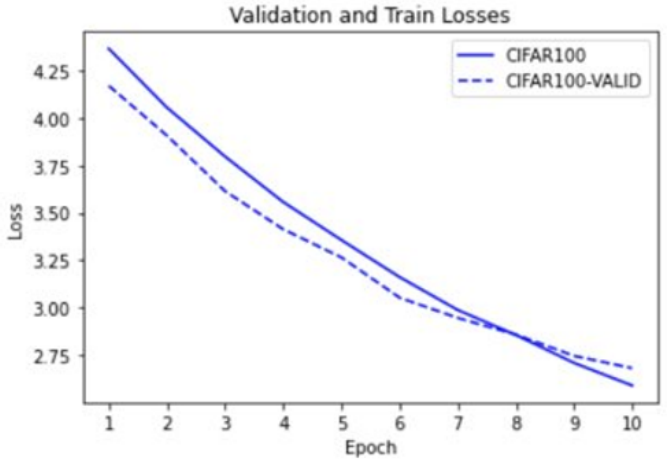

Fig. 11. Training and Validation Loss for CIFAR100 using AlexNet

C. VGG-16

VGG-16 is one of the most recognized models in the field of computer vision.VGG-16 is one of the major innovations that paved the way for other breakthroughs in this industry. VGG stands for Visual Geometry Group and 16 states it has 16 layers. The VGG-16, created as a deep neural network, outperforms benchmarks on a variety of tasks and datasets outside of ImageNet. It also remains one of the most often used image recognition architectures today. It was proposed by A. Zisserman and K. Simonyan from the University of Oxford. [11]

VGG-16 consists of 16 layers, of which 13 happens to be convolution layer and 3 are fully connected layers. It was designed for an input image of size 224X224. The convolution layers have 1X1 convolution filters for linear transformation of inputs, which also consists of a 3X3 sized receptive field. The stride is fixed to be 1 and it uses the Relu function as the activation function. The hidden layers also use the same activation function. The first two neural networks have 4096 neurons while the last one has 1000 neurons, with a softmax layer. The complete architecture of VGG-16 is huge, and respecting the constraints of the survey paper, we are not including the architecture visual format. [10]

VGG-16 preforms very good, on MNIST and Fashion-MNIST. This is evident from the training and testing loss graph(Fig - 12).

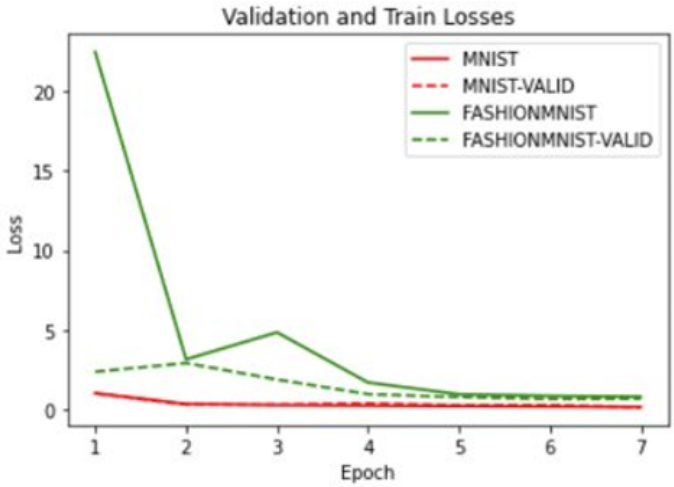

Fig. 12. Training and Validation Loss for MNIST Fashion-MNIST using VGG-16

VGG-16 shows a rapid decline in training and validation loss for CIFAR-100, but it seems not so competitive. It may be due to the environmental constraints we set for each of the models. If it were to be given more epochs to be trained upon, it may have performed well looking at the training and validation loss graph(Fig - 13).

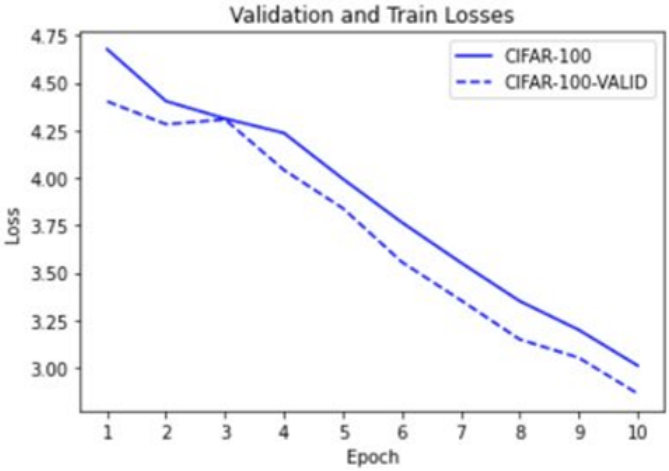

Fig. 13. Training and Validation Loss for CIFAR-100 using VGG-16

D. New Architecture 1

This architecture is exactly similar to Alexnet but, here we introduce the concept of Batch Normalization. Normalization

is used generally to bring all the features to the same scale, so the significant contribution of each feature is judged equally, without giving extra credits to any feature due to its higher range of values. Batch Normalization is adding extra layers to the deep neural networks to standardize the generated output in the hidden layers and keep the model stable. The word stable here signifies, the removal of the reducing/exploding gradient problem. Due to an increase in the number of layers, at certain times in deep neural networks, the gradient is significantly reduced or increases with great momentum. Any of these issues will disturb the whole network. Here, batch normalization comes into the picture. This layer standardizes and normalizes the input from the previous layer and forwards it to the next layer.

Batch normalization works in two ways:

1. Normalization of input.

This process' objective is to reduce the mean to 0 and set standard deviation to

a. Mean is calculate by the following formula:

$$\mu = \frac{1}{m} \times (\Sigma h_i)$$

where m is the number of neurons in layer h and h summation is batch input from the layer h. b. Standard deviation is given by:

$$\sigma = \frac{1}{m}(\Sigma(h_i - \mu)^2]^{1/2}$$

c. We use these values to normalize the hidden activation as follows:

$$h_i(norm) = (h_i - \mu)/(\sigma + \epsilon)$$

where ϵ is smoothing term. It comes handy if anytime the equation is reduced to division by 0.

2. Rescaling of Offset

Here the re-scaling and offsetting of the incoming input feature map takes place. There are two parameters for this, that is γ and β.

$$h_i = \gamma \times h_{i(norm)} + \beta$$

here γ is for re-scaling and β is for shifting of vector containing values from previous operations. [9]

Batch Normalization speeds up the training and handles internal covariate shifts. By applying this we ensure that the input to the next layer is distributed around the same mean and standard deviation. We applied batch normalization in the second layer, as here the input feature map must have passed through two convolution layers and one overlapping max-pooling layer. Now normalizing the inputs for the next convolution layer makes more sense, as there might be a loss of data or an addition of noise. Similarly, we added batch normalization after the last convolution and max-pooling layer, so each neuron of the fully connected layer would get the same distribution of data. The results were good.

As from the Fig 14 we can see the modified AlexNet performs excellent with MNIST and Fashion-MNIST.

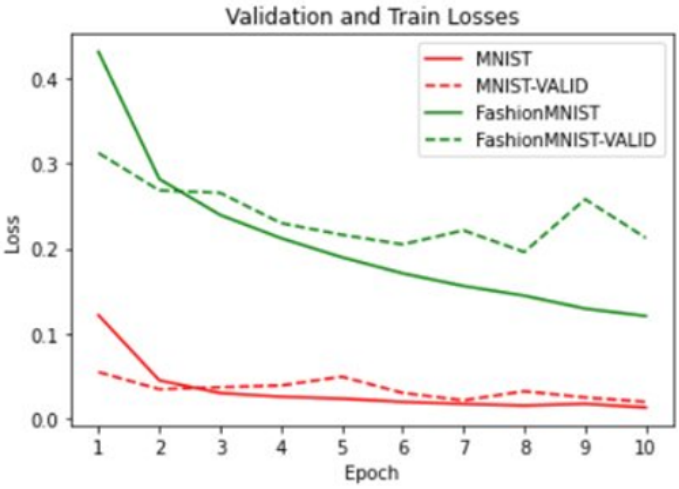

Fig. 14. Training and Validation Loss for MNIST Fashion-MNIST using New Architecture-1

The interesting part of the architecture is it performs well on CIFAR-100 too. We can see from Fig 15 that, while AlexNet was performing no so well for the same epoch and batch size, the New Architecture - 1 performs well on CIFAR-100 with the same given constraints.

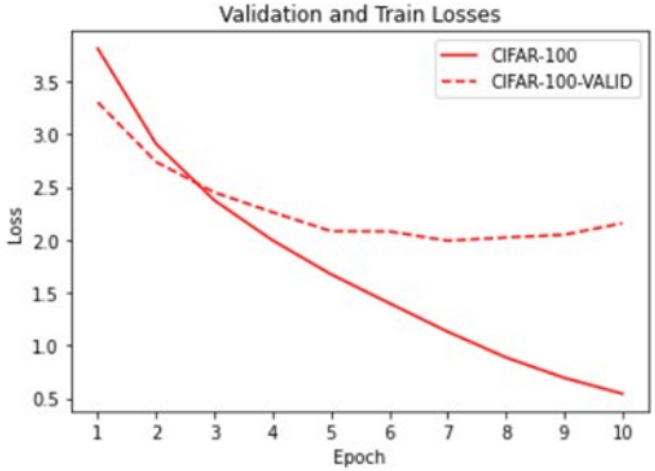

Fig. 15. Training and Validation Loss for CIFAR-100 using New Architecture-1

E. New Architecture 2

From the results of VGG-16 and other classical models, we noticed that such a deep architecture was not required for simple and relatively easy-to-learn datasets such as MNIST and Fashion-MNIST. To maintain the same level of accuracy a model with significantly lower complexity will suffice. We also wanted to drastically reduce the painfully long training time of VGG-16. We started focusing on reducing the number

of learnable parameters by removing many convolution layers. We noticed that stacking two convolutions 3x3 layers was very effective in learning features. A-Max pool layer was used in between the pairs of convolution layers. This reduced the overall dimensions of the input image thereby reducing the number of parameters.

From the previous architecture's improved performance we saw the importance of batch normalization and added it after every convolution. We also experimented with different activation functions and chose ELU (Exponential Linear unit). It showed significant improvement in speed and accuracy when compared to ReLU in similar experiments. Like ReLU and leaky-ReLU ELU also evades the vanishing gradient problem [13].

Training results

As expected, we saw a drastic reduction in the training time. The performance remains unhinged with constantly improving accuracy for both MNIST and Fashion-MNIST datasets.

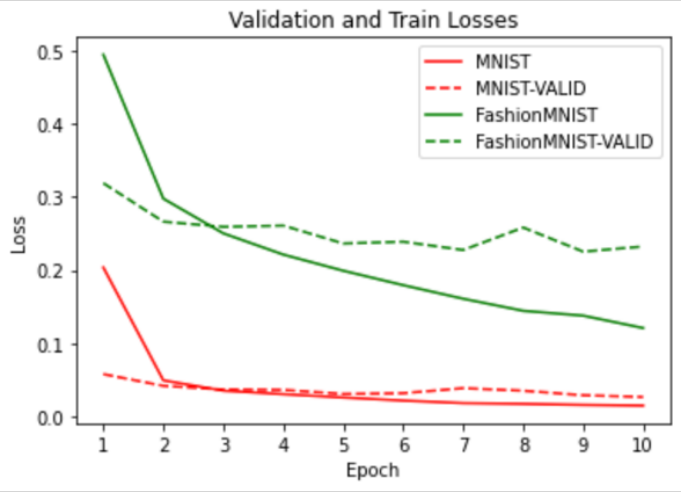

Fig. 16. Training and Validation loss for MNIST & FashionMNIST using New Architecture 2

The performance with the CIFAR-100 dataset was also on par with other deep classical models even with a reduced number of layers. The gain in accuracy could be attributed to batch normalization and the new activation function used.

VII. Modern Architectures

A. ResNet18

ResNet, also known as residual networks, uses batch normalization and skip connections to train deep layers without sacrificing performance. ResNet is similar to a VGG network that uses a sequential approach. The ILSVRC-2015 competition featured deep networks with more than 100 layers, and ResNet came out on top. The vanishing gradients problem[2], which results in relatively small derivatives, made it difficult to train CNN after VGG nets, which stacked up numerous layers going deep. As a result, skip connections—also known as

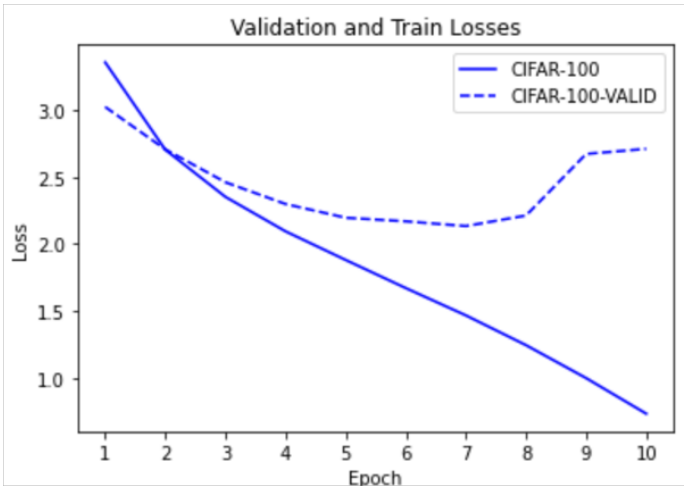

Fig. 17. Training and Validation loss for CIFAR-100 using New Architecture 2

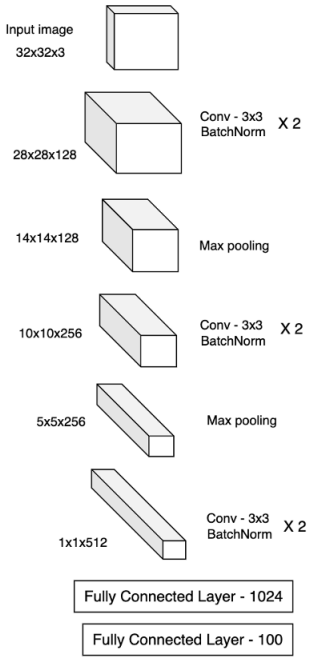

Fig. 18. New Architecture 2

residual blocks—came into being as a solution to the vanishing gradient problem and are used by ResNet design.

The figure 15 shows skip connections used in ResNet architecture. Skip connection is a process of mapping the

If X is the input and F(X) is the output from the layer, then the residual block output can be calculated as,

$$Y = F(x) + x$$

ResNet-18 is a CNN that is trained on more than a million images from the ImageNet Database. ResNet-18 can classify images into 1000 object categories. The architecture consists

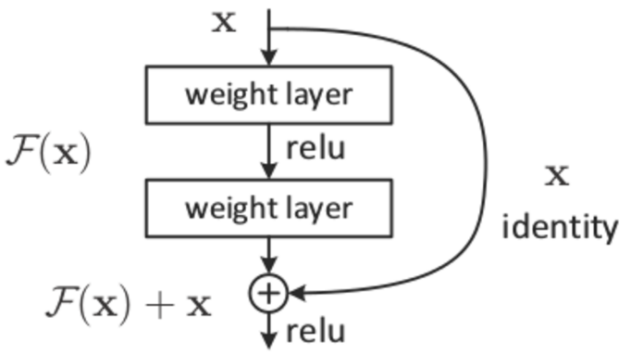

Fig. 19. ResNet Skip Connection [15]

of about 18 layers and the image input size of the network is 224×224.

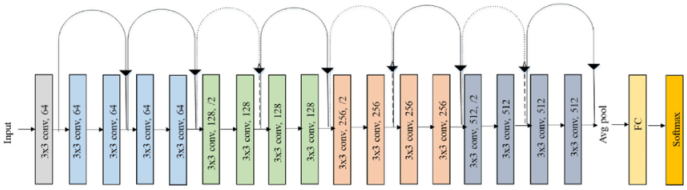

Fig. 20. ResNet-18 Architecture

The architecture of ResNet-18 comprises of the original model with 18 layer. It uses filters mostly of size 3X3, and does downsampling with stride set to 2 for each convolution layer. It uses residual learning block, and those are of two types. Either for the case where the input and output dimesions are same or for the case when they both are not same. Each module of the ResNet model contains four convolution layers excluding the input layer. The original architecture explained as the figure.20

The loss for MNIST and FashionMNIST datasets is very low from the beginning of training which is also reflected in the training accuracy of the model. The use of skip connections made a significant improvement in the training time.

The training accuracy did not improve much with the current hyperparameters used to compare the models. But the loss can be seen reducing gradually when it reaches the last epoch. Increasing the epoch size would improve the training accuracy and thus can further reduce the training and validation losses .

B. Inception-v1 (GoogLeNet)

The advent of Lenet and AlexNet was a breakthrough in the study of convolution neural networks in the domain of image classification. It laid the foundation for more advanced and deeper neural networks. Naturally, to learn a complex dataset with a large corpus of images with multiple classes, researchers would often expand the Lenet/AlexNet implementation to design a deeper neural net with multiple

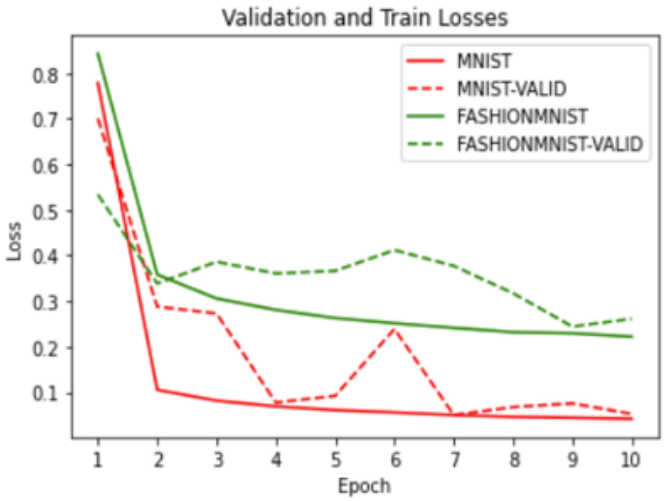

Fig. 21. Validation and training loss for MNIST and FashionMNIST using ResNet-18

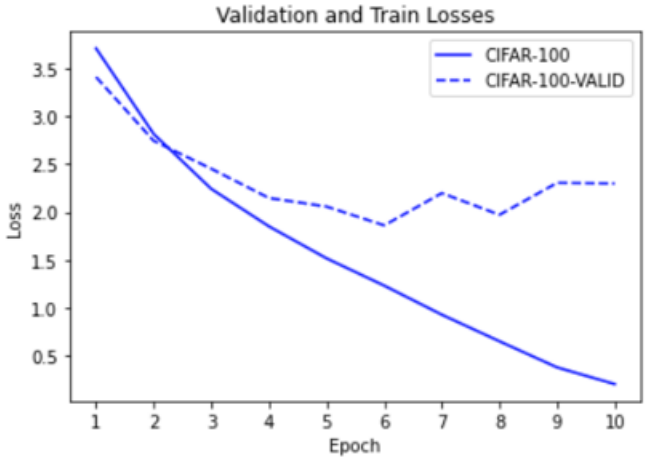

Fig. 22. Validation and training loss for CIFAR-100 using ResNet-18

stacked convolution layers. This led to a significant performance gain. However, increasing the layers to create more extensive networks came at a cost. Large networks are prone to overfitting and suffer from either exploding or vanishing gradient problems. Additionally, the computational costs were so high and spending a lot for a small performance gain seemed highly impractical.

GoogLeNet, also known as Inception-v1, was a deep convolutional neural network architecture that was developed by researchers at Google in 2014. The motivation behind the project was to solve most of the above-mentioned problems of traditionally developed large convolution neural networks. It did so by using a technique called ”Inception”. It involved using multiple parallel convolutional layers with different filter sizes, which allowed the network to learn more complex features from the input data. The GoogLeNet was trained on the ImageNet dataset, which consists of millions of labelled

images from a wide range of categories. After training, the network was able to achieve impressive performance on the image classification task, and it won the ILSVRC 2014 competition with a top-5 error rate of 6.7%.

The Inception module

The usage of convolution layers with pooling layers proved very effective in identifying distinguishing features. While developing a traditional neural network one must often choose one layer over the other and should accept the trade-off that comes with choosing between different filter sizes. For example, the 1x1 filter is useful in dimensionality reduction and learning cross-channel patterns whereas the 3x3 and 5x5 layers can effectively identify spatial patterns [5]. The Pooling layer reduces the dimensions of the input data. The inception module aims to bring all these benefits without having the trade-off of choosing one over the other. It combines all the mentioned layers and computes them in parallel. The inception module contains convolution layers of filter sizes 1, 3, and 5 with a max pool layer. The outputs are combined in the concatenation layer. ReLU activation function is used in the convolution layer.

In a naive inception module, the max-pooling layer is performed with a padding value set to the same to maintain the same height and width as the other (feature maps) of the convolutional layers within the same Inception module. But doing all the operations mentioned above will come with a huge computational cost. It can quickly increase the number of channels and can be a hindrance in reaping benefits from the increased representational power.

To overcome this the naive inception block is replaced with an inception block with dimensionality reduction. The idea is to perform a 1x1 convolution before the 3x3 and 5x5 convolutions which reduce the dimension of the input data.

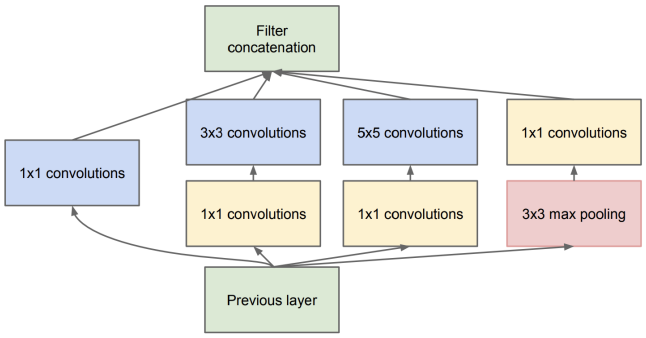

Fig. 23. Inception Module [2]

As a result of replacing traditional convolution layers with the inception module gives the ability to extract features from input data at varying scales through the utilisation of varying convolutional filter sizes. Also, the modified inception module leads to efficient utilisation of computing resources.

Architecture

Our implementation of the GoogleNet followed the original version which had 22 layers with 9 Inception blocks. Since the architecture is quite deep and large, the pictorial representation has been skipped.

Training results

For MNIST and FashionMNIST the losses started to converge right from the initial epochs and continued to fall, and signs of overfitting was controlled by our model selection criteria. The accuracy for both the data sets remained high as expected.

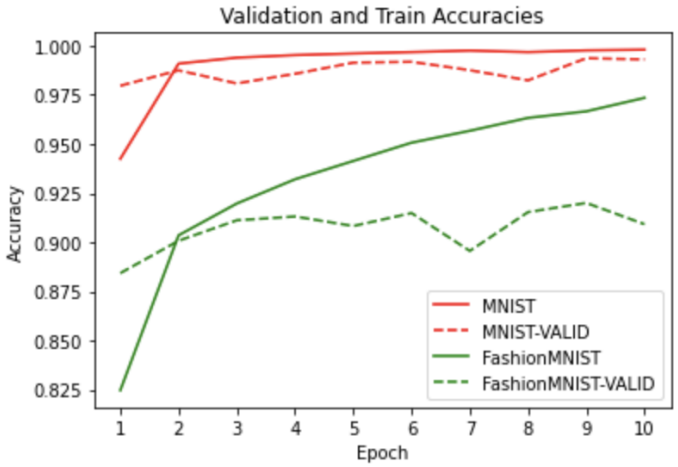

Fig. 24. Training and Validation accuracy for MNIST & FashionMNIST using Inception-v1

For CIFAR-100 although the losses did not come down as low as the other datasets, the losses were lowering and would have continued to downtrend for future epochs as well.

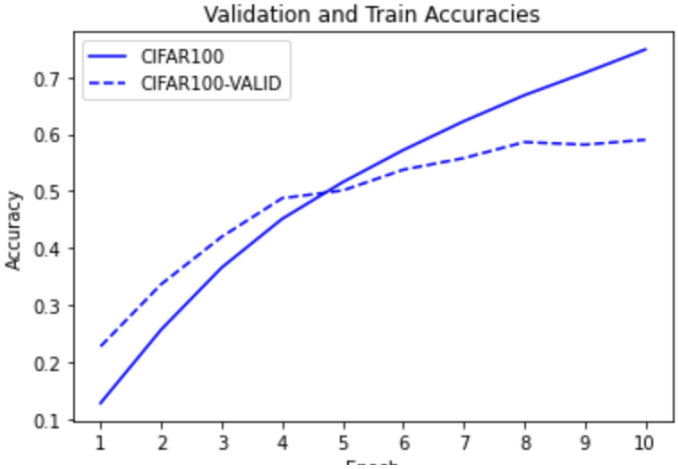

Fig. 25. Training and Validation accuracy for CIFAR-100 using Inception-v1

C. New Architecture 3

From the above results, we observed that the inception block from the GoogLeNet was very effective. It was evident

from its performance on the CIFAR100 dataset. It posted the lowest train and validation loss and was the best-performing model. The idea of the inception block was very intuitive and we wanted to design a new architecture around it. We also observed that it was computationally heavy which resulted in training times being higher than that of some classical architecture. To address this, we halved the number of inception modules from 9 to 4 and sandwiched them between normal 3x3 convolution layers. Drawing ideas from the previous success we changed the convolution layer to use the ELU activation function. Refer Fig. 21 for the complete architecture

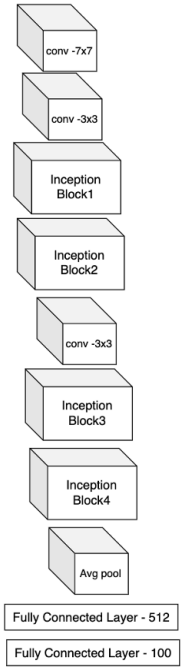

Fig. 26. New Architecture 3

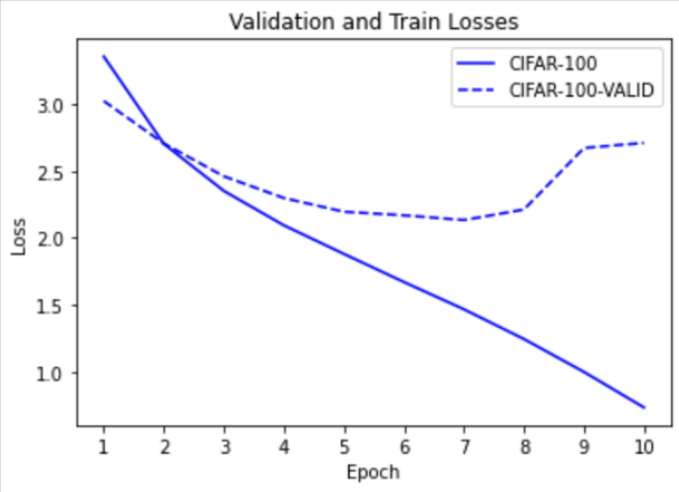

Fig. 27. Validation and training loss for CIFAR-100 using New Architecture 3

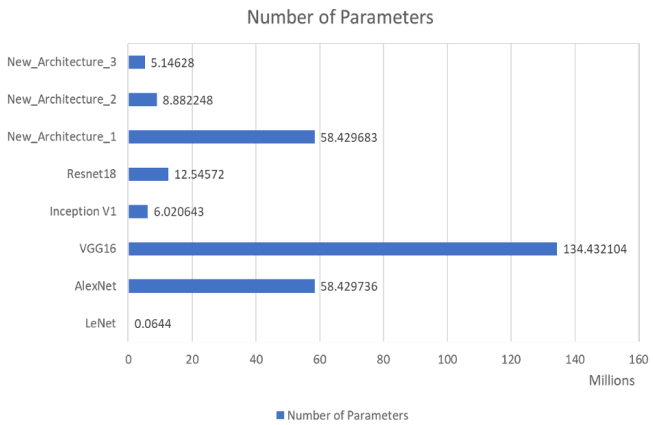

Fig. 28. Trainable parameters of all the models

Training results

As with any other modern architecture, this model did not find it difficult to learn MNIST and Fashion-MNIST datasets. We saw a slight dip in performance with the CIFAR100 dataset when compared to the original Inception-v1 that could be attributed to the reduction in the total number of inception layers. But we were able to achieve the purpose of creating this model, ehich was to reduce the computational complexity. The training time were consideribly lower than the original model.

VIII. Results and Analysis

Even before the initial model was coded and our training algorithms were run, we predicted which datasets will perform well with certain models and which will not, through exploratory data analysis. The results we obtained convey a very similar message. To effectively learn complex data sets like CIFAR100, we would require a very deep and unconventional model.

Comparing all the classical architectures, it was surprising that a simple 5-layer architecture like LeNet could learn the MNIST dataset with near-perfect accuracy and progress well with Fashion MNIST. Even if its performance was abysmal with the CIFAR100 dataset, it is the fastest model to train among all and the one with the least number of parameters to learn. This shows that LeNet can still be used to classify greyscale images with well-defined features.

AlexNet and VGG16 which had much more complex layers did not show any notable improvements with the grey-scale datasets. But the added complexity improved the accuracy of the 3 channel image. Although, AlexNet would still not suit a vague dataset like CIFAR100. VGG-16 showed a strict downward trend in training and validation losses, but the training was stopped due to environmental and experimental constraints. It was the most computationally complex model with more than 130 million parameters and understandably, it took a very long time to train.

Our models which were either modifications or extensions

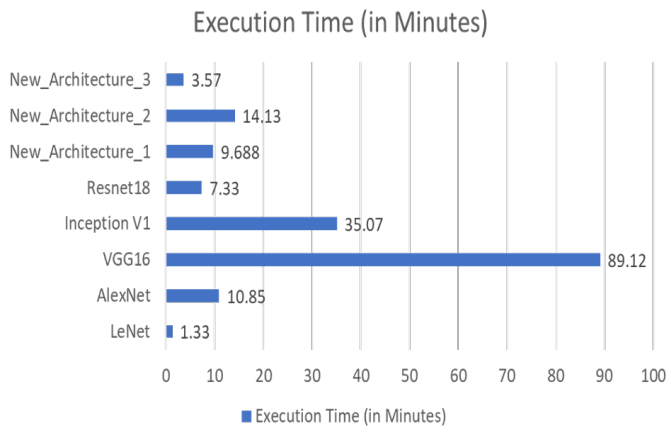

Fig. 29. Training time for all the models

of the classical architectures showed improvement in accuracies while being computationally less complex. Simple yet powerful concepts such as Batch Normalization and ELU activation function made the model capable of outperforming other complex models. Just to compare the efficiency, our models completed the training for the entire dataset before VGG-16 could complete a single epoch of training.

Modern architectures introduced new methodologies like residual and inception blocks, Which made the model more complex without it being deeper. Variations of these models are the state of the art in image classification, and it showed in the final results. The inception-v1 was not the quickest but outperformed all models purely in terms of the error rate. ResNet offered a good balance between performance and computation complexity. The new architecture based on inception delivered similar performance but only required half the layer of the original inception module.

Model \ Dataset	MNIST	F-MNIST	CIFAR-100
LeNet	0.9886	0.8955	0.2699
AlexNet	0.9741	0.7931	0.3309
VGG-16	0.9885	0.8955	0.2903
New Arc 1	0.9946	0.9246	0.5039
New Arc 2	0.9888	0.9117	0.4538
ResNet-18	0.9868	0.9030	0.5035
Inception-V1	0.9939	0.9114	0.5961
New Arc 3	0.9931	0.9191	0.4681

IX. Conclusion and Future Work

Overall, this study was focused on understanding the success of Convolution Neural Networks in image classification problems. We analyzed various layers that make up a highly accurate model by comparing the performance of various classical and modern architectures that set the standard for this domain. We investigated the importance of techniques such as skip connections, inception and batch normalization and their impact on the overall performance of the model. This led to the design of three new architectures, which showed promising results. While it was extremely challenging to implement such large architectures and train them in commodity hardware, we arduously made sure that we contained the experimental bias as much as possible. Finally, we made a detailed comparison of the performance of these modules using various metrics such as loss, accuracy, and run-time.

In our Future studies, we would like to experiment with these models using a varying set of hyperparameters with effective feature reduction. We would also compare deeper models and their performance using parallel computing and high-performance GPUs.

References

[1] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

[2] C. Szegedy et al., 'Going Deeper with Convolutions', in Computer Vision and Pattern Recognition (CVPR), 2015.

[3] L. Peng, B. Qiang and J. Wu, "A Survey: Image Classification Models Based on Convolutional Neural Networks," 2022 14th International Conference on Computer Research and Development (ICCRD), 2022, pp. 291-298, doi: 10.1109/ICCRD54409.2022.9730565.

[4] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[5] Blog: Deep Learning: Understanding The Inception Module, https://towardsdatascience.com/deep-learning-understand-the-inception-module-56146866e652

[6] Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

[7] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[8] Zhang, H., Chen, W., & Liu, T. (2018). Train Feedfoward Neural Network with Layer-wise Adaptive Rate via Approximating Back-matching Propagation.

[9] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[10] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.

[11] Tammina, Srikanth. (2019). Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. International Journal of Scientific and Research Publications (IJSRP).

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 1097–1105.

[13] Z. Qiumei, T. Dan and W. Fenghua, "Improved Convolutional Neural Network Based on Fast Exponentially Linear Unit Activation Function," in IEEE Access, vol. 7, pp. 151359-151367, 2019, doi: 10.1109/ACCESS.2019.2948112.

[14] Blog: Deep Convolutional Neural Networks: https://www.run.ai/guides/deep-learning-for-computer-vision/deep-convolutional-neural-networks

[15] Blog: Residual Neural Network:https://iq.opengenus.org/residual-neural-networks/