Deeper Networks for Image Classification

Hussain Alkhorsan
School of Electronic Engineering and Computer Science
Queen Mary University
London, UK
ec211157@qmul.ac.uk

***Abstract*—The purpose of this work is to study several state-of-the-art deep neural network architectures for the task of image classification on popular image datasets, and to provide a critical evaluation of their performance through several experiments. The architectures studied in this paper include VGG-11, ResNet34, GoogLeNet and InceptionV3. The datasets used to train and evaluate the architectures include MNIST, FashionMNIST and CIFAR10. Furthermore, later experiments show that applying batch normalization and data augmentation techniques significantly improves model performance in most networks by reducing the risk of overfitting.**

I. Introduction

Deep neural networks have seen tremendous success in recent years particularly in the computer vision domain. Since the introduction of Convolutional Neural Networks (CNNs), several architectures have been proposed to extract complex features from an image through learnable weights. Prior to CNNs, traditional computer vision techniques often required hand-crafted features using various image processing algorithms to extract important information from an image. However, this approach was not scalable due to the amount of effort required in feature extraction. Recent advancements in computer hardware and the availability of large amounts of data have replaced the need for hand-crafted feature engineering by training a CNN to adaptively learn a hierarchy of features through backpropagation using multiple building blocks.

The architecture of a CNN is heavily inspired by the structure of the animal visual cortex. The building blocks of a CNN include convolutional layers, pooling layers and fully connected layers. The convolutional layer slides a filter or kernel over the input image and performs low-level feature extraction resulting in a feature map that is fed to the next convolutional layer in order to extract higher-level features. The objective of the pooling layer is to reduce the spatial size of the features from the convolutional layer by reducing the number of parameters and computation in the network. Each pooling layer operates independently on each feature map and are usually added after the convolution operation. Stacking multiple convolutional layers allows the network to learn complex non-linear functions from an input image which are then fed to the fully connected layer that outputs probabilities for each class in the case of image classification. The classification errors from the output are then backpropagated across the network to update the weights and this process is repeated over several iterations.

Image classification is the process of associating a given image with a class label and has been found to be useful in a wide variety of domains such as medical imaging, traffic control systems and robotics. CNNs for image classification are trained using supervised learning on large amounts of labelled data. The most popular database for benchmarking computer vision algorithms is ImageNet which contains millions of labelled images and holds an annual competition known as ImageNet Large Scale Visual Recognition (ILSVRC) to record state-of-the-art performances.

II. Related Work

CNNs have gone through rapid evolution since their first introduction in the 1980s. The earliest known CNN architecture is Neocognitron [1]. which was heavily inspired by the work of Hubel and Wiesel [2] [2]. on the neural basis of visual perception. The Neocognitron introduced the two fundamental building blocks of CNNs: the convolutional layer and downsampling layer. Although CNNs were invented in the 1980s, their breakthrough did not come until the early 2000s when recent advancements in computer hardware became much more accessible. The idea behind training CNNs using gradient descent was first proposed by Yann LeCun et al. [3]. in their work on LeNet-5 which used backpropagation to learn convolution kernel coefficients directly from images to recognise hand-written digits. This approach gained considerable interest and formed the foundation of modern computer vision. Recent developments have shown that the depth of a CNN is of crucial importance in improving model performance. In 2012, AlexNet [4]. which borrowed ideas from Yann LeCun et al. became the first CNN consisting of 8 layers to win the ILSVRC outperforming previous competitors by a large margin [5]. The success of AlexNet inspired more research into deep neural networks, [6]. introduced VGG – using a network similar to AlexNet but with 16-19 layers and 3x3 convolution filters which also gained considerable interest due to its ability to generalize well to other datasets.

However, as models become increasingly deep, they become more difficult to optimize due to the problem of vanishing gradients [7]. This was addressed in [8]. by using normalized initialization which allowed networks with many layers to converge using stochastic gradient descent with backpropagation [9]. Another problem arises known as the degradation problem when training accuracy becomes saturated and leads to high training errors. This was due to the difficulty in propagating

information from shallower layers as networks get deeper. The degradation problem was explored in [10]. by using a deep residual learning framework known as ResNets. Instead of learning the underlying mapping, the network fits a residual mapping by adding skip connections which allow information to flow more easily from one layer to the next.

Alternative architectures were also explored to tackle the problem of overfitting and exploding or vanishing gradients which were very common in deeper networks. In 2014, Google introduced the Inception network [11]. and a variant of it known as GoogLeNet. The main contribution of their work was to use inception modules to reduce the cost of computation by using kernel filters of different sizes at the same level instead of stacking them sequentially. As a result, by performing convolutions on the same level, the network gets progressively wider instead of deeper. The use of inception modules became widely adopted which led to a series of improvements in [12] and [13].

III. Method

VGG-11

The VGG-11 model is a deep neural network with 11 weighted layers and (3x3) convolution filters. The upper two fully-connected layers have 4096 channels and the third fully-connected layer has 10 channels where each channel corresponds to one target class. Max-pooling with a (2x2) kernel is applied after each convolutional layer to extract a feature map containing the most prominent features. The first convolutional layer takes in as input either 3 input channels for RGB images or 1 input channel for grayscale images depending on the dataset used. Dropout is applied in the fully connected layers to reduce overfitting on the training data with a probability of 0.5 and softmax is applied at the end to convert the output into probabilities. In total, the model has 128,807,306 parameters.

ResNet-34

The ResNet architecture used in the experiments contains 34-layers with (1x1) and (3x3) convolution filters in each layer. Batch normalization is also applied after each convolutional layer. The last layer includes one fully connected layer that has 10 channels for each class. In this model, Dropout was not applied to keep the architecture similar to the original paper as the authors mention no usage of dropout in their work. Similar to VGG-11, the input takes in either 1 channel or 3 depending on the dataset used. Each layer in the architecture forms a residual block where each block can be defined as module with batch normalization and ReLU activation applied after each convolutional layer. In total, ResNet34 has 21,289,802 parameters which is approximately 6 times smaller in magnitude than the VGG-11 network.

GoogLeNet

The GoogLeNet architecture contains 22 layers with each layer containing an inception module of a fixed convolution size. In each inception module (1x1), (3x3) and (5x5) convolution filters are applied with a (3x3) max pooling. The outputs are then stacked together to produce a final output. The Inception architecture also requires a different training strategy than the other models by additionally training auxiliary classifiers. These are intermediate classifier branches consisting of a (5x5) average pooling layer with a stride of 3 and (1x1) convolutions with 128 filters, 2 fully connected layers with 1024 and 1000 outputs respectively. The generated loss from these layers are added to the total loss with a weight of 0.3. In the experiments, the last fully connected layer is modified to produce 10 outputs with respect to the number of classes in the datasets. Dropout with a probability of 0.4 is also applied to the main output and dropout with a probability of 0.7 is applied to the auxiliary outputs from the Inception blocks. The GoogLeNet architecture has in total 10,344,814 parameters.

InceptionV3

The InceptionV3 network is heavily inspired by the GoogLeNet/InceptionV1 architecture but with some considerable improvements. Upon realizing that the auxiliary classifiers did not contribute much until the end of the training process when model accuracy began to saturate, the authors of InceptionV3 argued in their paper that that the auxiliary classifiers could function as regularization techniques due to batch normalization and dropout operations [12]. The other modifications described in the paper in addition to the utility of the auxiliary classifiers are factorization into smaller convolutions, spatial factorization into asymmetric convolutions and grid-size reduction. InceptionV3 has 48 layers and a total of 24,371,444 parameters roughly 2.5 more than the original GoogLeNet/InceptionV1 architecture.

IV. Datsets

MNIST

The MNIST dataset [14]. is a subset collection of the larger NIST database of handwritten digits which consist of a training set of 60,000 examples and a test set of 10,000 examples. The dataset is size-normalized with each example having a 28x28 size dimensions. The MNIST dataset is used for the purpose of classifying digits from 0-9 written in different styles.

Fig. 1. The MNIST database of handwritten digits.

FashionMNIST

The FashionMNIST dataset [15]. is a dataset of Zalando's article images, similar to MNIST it contains a training set of 60,000 examples and a test set of 10,000 examples of 28x28 greyscale images. FashionMNIST serves as an alternative replacement of the original MNIST dataset for benchmarking computer vision algorithms. The dataset contains 10 class labels of fashion categories.

Fig. 2. The FashionMNIST database of different fashion categories.

CIFAR-10

The CIFAR-10 [16] collection is a dataset of 60,000 images with 50,000 training examples and 10,000 test examples and is one of the most widely used datasets for computer vision research. It is a subset of the 80 Million Tiny Images database but with independently generated labels [17]. Similarly to the previous datasets, it contains 10 class labels of different object categories and each example contains a 32x32 RGB colour image with 3 channels.

Fig. 3. The CIFAR-10 datasbase of different object categories.

V. Experiments

Training strategy

The training strategy in each of the experiments was kept the same to evaluate the baseline architectures however, some modifications in additional experiments were applied to see whether they contributed to an improvement in model performance. Each model was trained for 25 epochs with a batch size of 64 and using the Adam optimizer with a learning rate of 0.001. The loss function used in the experiments was the cross-entropy loss criterion which measures the difference between two probability distributions. Due to the amount of convolution operations, the CIFAR-10 dataset had to be resized to dimensions 224x224 and similarly the MNIST and FashionMNIST datasets were resized to 227x277 dimensions.

Testing results

VGG-11

The VGG-11 model achieved a 98.88% accuracy on the MNIST dataset after 25 epochs. However, results from the experiments show that the model managed to achieve very high accuracy after the first epoch and only modest improvements in subsequent iterations. Furthermore, the model managed to converge after 6 epochs and there was no substantial improvement for the remaining of the training duration. VGG-11 manged to achieve an 88% accuracy on the FashionMNIST dataset however, the model was likely overfitting the training data due to the validation loss increasing after 8 epochs. During experimentation on the CIFAR-10, VGG-11 achieved an accuracy of 75.2% with little improvement on the validation loss. Fig 4. shows a visualization of the training and validation graphs for each experiment MNIST (blue), FashionMNIST (red) and CIFAR-10 (brown).

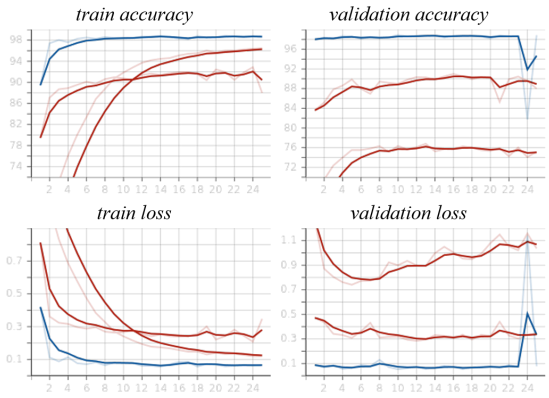

Fig. 4. VGG-11 - training and validation metrics

ResNet-34

ResNet-34 achieved a 99.51% validation accuracy on the MNIST dataset. There seemed to be no drastic improvement after subsequent training iterations with a 97.94% accuracy being achieved after one training iteration. It is worth noting that most standard implementations of neural networks achieve an accuracy at this range due to the quality of the MNIST dataset. However, although not a significant difference, ResNet-34 managed to exceed VGG-11 validation accuracy by 0.63%. Furthermore, on the FashionMNIST dataset, ResNet-34 achieved a 93.02% accuracy and an 85.24% accuracy on CIFAR-10. Fig 5. shows visualization graphs for each of the ResNet-34 experiments MNIST (blue), FashionMNIST (brown) and CIFAR-10 (red). The validation loss graph

suggests the model experienced significant overfitting on the CIFAR-10 dataset.

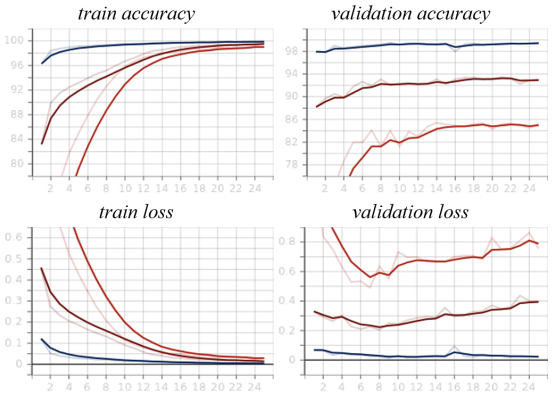

Fig. 5. Resnet-34 - training and validation metrics

GoogLeNet

The GoogLeNet network achieved a 99.49% accuracy on the MNIST dataset and also a 93.44% and 85.74% on the FashionMNIST and CIFAR-10 datasets respectively. Fig 6. shows a visualization from the experiments. The validation loss on the FashionMNIST (brown) and CIFAR-10 (red) experiments showed similar patterns. After 8 training epochs, they both begin to increase suggesting they are overfitting. Furthermore, the graphs show the validation loss decreasing uniformly for the MNIST experiment (blue) and begins to stagnate after approximately 4 epochs. There does not seem to be any overfitting on the MNIST experiment.

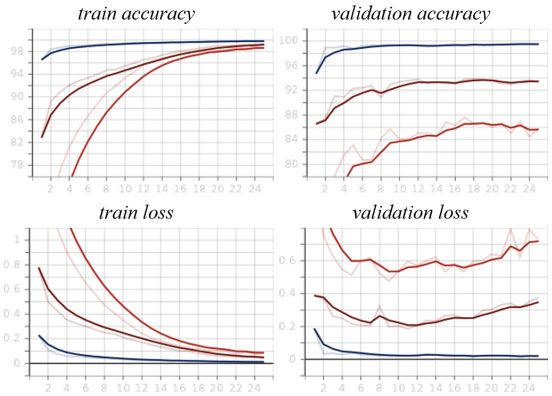

Fig. 6. GoogLeNet - training and validation metrics

InceptionV3

The InceptionV3 model showed good performance across all experiments, achieving an accuracy of 99.51%, 94.40% and 87.78% on the MNIST, FashionMNIST and CIFAR-10 experiments respectively. Fig 7. shows the validation loss across all experiments MNIST (blue), FashionMNIST (brown) and CIFAR-10 (red) showing similar patterns and began to stagnate after approximately 4 training epochs.

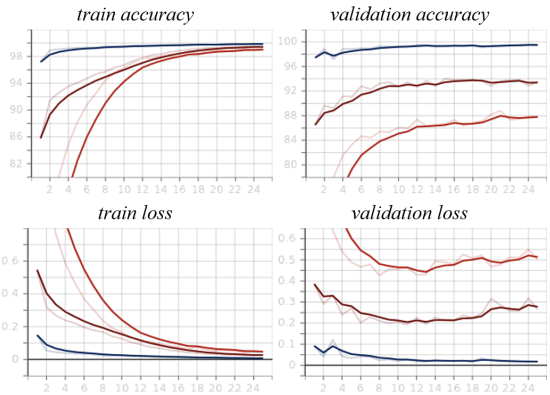

Fig. 7. InceptionV3 - training and validation metrics

Further evaluation

VGG-11

In the further experiments, the modifications were evaluated using the CIFAR-10 dataset since this was where the networks were usually overfitting. Batch normalization and data augmentation was applied to the VGG-11 network which managed to reduce overfitting. Examples of some of the augmentations applied were to use random resized crops and random horizontal flips. The resized crops were resized back to the original 224x224 dimensions and flips were applied with a probability of 0.5. Performing further data augmentations and significant alterations on the training dataset could diverge the model from learning the ground truth which may lead to bad performance on the validation set therefore, the number and type of augmentations applied were limited. Applying batch normalization also increased the time taken to train the model. Fig 8. shows a comparison of the validation loss when applying the modifications (blue) with respect to the original baseline (red).

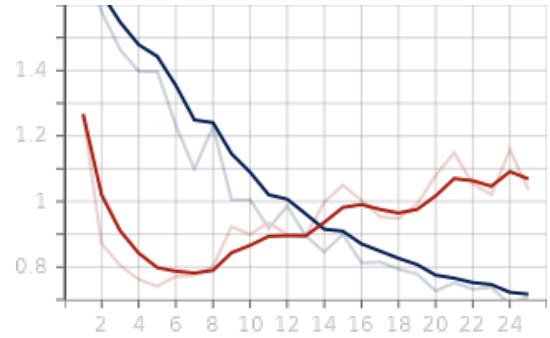

Fig. 8. VGG-11 - validation loss graph

ResNet-34

In this experiment, the optimizer was changed to Stochastic Gradient Descent (SDG) instead of Adam which was used in the baseline experiments. Furthermore, data augmentation was also applied to the training data. Fig 9. shows the validation

loss after applying the modifications (blue) and the original baseline (red). The modifications managed to mitigate the overfitting that occurred in the original baseline. The final validation accuracy achieved after applying the modifications was 77.94%

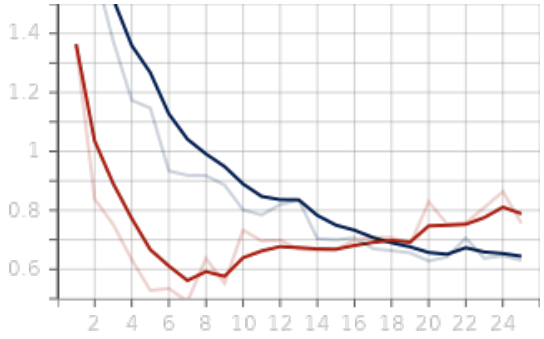

Fig. 9. ResNet-34 - validation loss graph

GoogLeNet

Applying data augmentations on the training dataset also managed to reduce overfitting in the GoogLeNet model. Fig 10. shows a comparison between the validation loss of the original baseline (red) compared to the model when applying data augmentations (blue). The validation seems to decrease uniformly after each training epoch whereas in the original baseline, the validation loss fluctuates. After epoch 21, the validation loss in the original baseline seems to begin increasing where as in the second experiment the validation loss seems to continue to decrease.

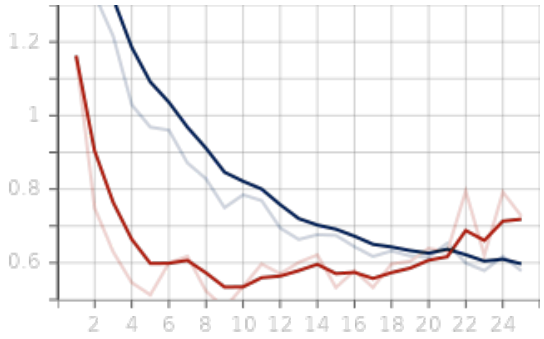

Fig. 10. GoogLeNet - validation loss graph

InceptionV3

Applying data augmentations to the InceptionV3 network did not seem to improve the performance of the model. The starting validation loss was higher than the initial validation loss in the original baseline. This may be due to the increase in variance of the dataset when applying the augmentations and thus may affect the networks' ability to generalize well. However, after 25 epochs, fig 11. shows the validation loss still decreasing when applying data augmentations (blue) in comparison to the original validation loss (red) which seems to stagnate and thus increasing the number of epochs may help bring the validation loss down. Furthermore, it should also be noted that the original InceptionV3 baseline model has already shown good performance across all datasets and further modifications could possibly increase the risk of overfitting.

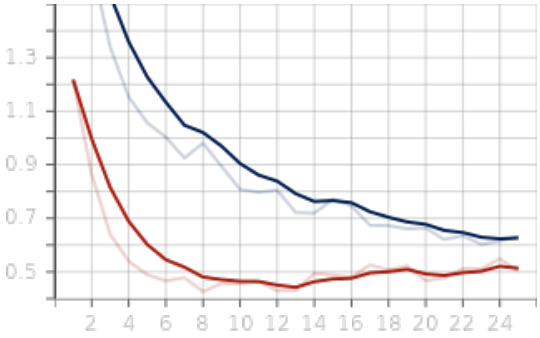

Fig. 11. InceptionV3 - validation loss graph

VI. Conclusion

Results from the experiment show that all models managed to achieve very high accuracy across all experiments however, all networks were extremely susceptible to overfitting. The problem of overfitting could be mitigated in many ways such as increasing the size of the dataset either by using data augmentation techniques as seen in the further experiments or using larger datasets. Dropout has also been shown as a powerful technique in tackling the problem of overfitting. In the baseline experiments, InceptionV3 managed to achieve the highest accuracy across all datasets in comparison to the other models with GoogLeNet achieving second place, ResNet-34 being third and VGG-11 achieving last place. Furthermore, the performance of ResNet-34 and GoogLeNet networks were extremely similar although having different architectures. ResNet-34 utilizes residual blocks while GoogLeNet utilizes inception blocks. In general both architectures seem to achieve great accuracy. Another important aspect worth mentioning is the time taken to train each network. The Inception-based architectures often took the longest to train due to number of layers and convolution operations required with ResNet-34 and VGG-11 often requiring shorter training times. However, this is not a good metric for evaluating the computational cost of the networks since the Inception architectures were trained on GPUs with higher RAM capacity and the ResNet-34 and VGG-11 architectures were trained using CPUs with lower RAM capacity. Due to the size of the Inception networks, there was difficulty in fitting them to memory using the less advanced computing resources and thus this should not be considered a good indicator on the computational time required in training the architectures. In summary, this work aimed to perform a critical evaluation on 4 state-of-the-art baselines in computer vision. Possible future work could be to extend this evaluation to other learning tasks. Deep neural networks have also shown state-of-the-art performance in other computer vision domains such as object detection, image

segmentation and video classification. Furthermore, there is a possibility in exploring other ways in improving the accuracy of the models. For instance, other works [18]. have shown using dynamic learning rates during various stages of the training process offers higher accuracy and faster convergence however, this has been left as a direction for future research.

REFERENCES

[1] Fukushima, K. 1980. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position.

[2] Lienhard, D.A. 2017. David H. Hubel and Torsten N. Wiesel's Research on Optical Development in Kittens.

[3] LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition.

[4] Krizhevsky, A.; Sutskever, I.; and Hinton, G.E. 2012. ImageNet Classification with Deep Convolutional Neural Networks.

[5] Tsang, S. 2018. Review: AlexNet, CaffeNet – Winner of ILSVRC 2012 (Image Classification).

[6] Simonyan, K. and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition.

[7] Glorot, X. and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks.

[8] LeCun, Y.; Bottou, L.; Orr, G.B.; and Muller, K.R. 1998. Efficient BackProp.

[9] LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E; Hubbard, W.; and Jackel, L.D. 1989. Backpropagation Applied to Handwritten Zip Code Recognition.

[10] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition.

[11] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going Deeper with Convolutions.

[12] Szegedy, C.; Vanhoucke, V.; Loffe, S.; Shlens, J.; and Wojna, Z. 2015. Rethinking the Inception Architecture for Computer Vision.

[13] Szegedy, C.; Loffe, S.; Vanhoucke, V.; and Alemi, A. 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.

[14] LeCun, Y.; Cortes, C.; and Burges, C.J.C. The MNIST Database of Handwritten Digits.

[15] Zalando Research. Fashion-MNIST: A MNIST-like fashion product database. Available at https://github.com/zalandoresearch/fashion-mnist

[16] Krizhevsky, A.; Nair, V.; and Hinton, G. The CIFAR-10 dataset. Available at https://www.cs.toronto.edu/ kriz/cifar.html

[17] Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

[18] Wu, Y.; Liu, L.; Bae, J.; Chow, K.; Iyengar, A.; Pu, C.; Wei, W.; Yu, L.; and Zhang, Q. 2019. Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks.