Image Classification of a Limited Dataset

Jacqueline Heaton

April 2021

Abstract

Image classification is an area of interest for many researchers, as it has a wide variety of applications. Many research projects require some form of image classification as a step in their overall goal, whether for diagnosis, object recognition, object localization, facial recognition, or something else. Due to the vast amount of research done on this kind of task, there are many different image datasets available, along with artificially intelligent models that have already been trained on an image dataset. The goal of this project is to explore the use of transfer learning when it comes to image classification, specifically, models trained on the ImageNet dataset when classifying images from the CIFAR-100 dataset. Residual Networks seem to preform the best, with more layers improving the accuracy. A basic ResNet50 was able to achieve 78% test accuracy with minor data augmentations, while the regular convolutional neural networks tended to get accuracies around 45%.

1 Introduction

Image classification has been an area of interest for many years, with models like AlexNet [9], VGG16 [11], and ResNet50 [5] drawing attention from researchers all over the world for their image classification accuracy. Since then, many improvements have been made, with models like the EffNet-L2 [8] and GPipe [6] achieving near perfect accuracy on the CIFAR-10 dataset. The CIFAR-10 dataset has 60,000 images equally separated into 10 classes, and is one of the baseline datasets to use for image classification. Since it only has 10 classes, is much easier to achieve good results on the CIFAR-10 dataset than the CIFAR-100 dataset, as the CIFAR-100 dataset has 100 classes, with the same number of images. In the CIFAR-100 dataset each class only has 600 images as opposed to the 6000 provided in the CIFAR-10 dataset, making it much harder to achieve a high accuracy, though there are still several models that manage to get a test accuracy above 90%. The goal of this project is to classify images from the CIFAR-100 dataset using pretrained convolutional neural networks (CNNs) and residual networks (ResNets), and to explore how both pretraining and using certain kinds of data augmentation affect the accuracy. This will ideally allow for an easy way to train a network for research use, rather than requiring a complex model to be implemented and trained from scratch.

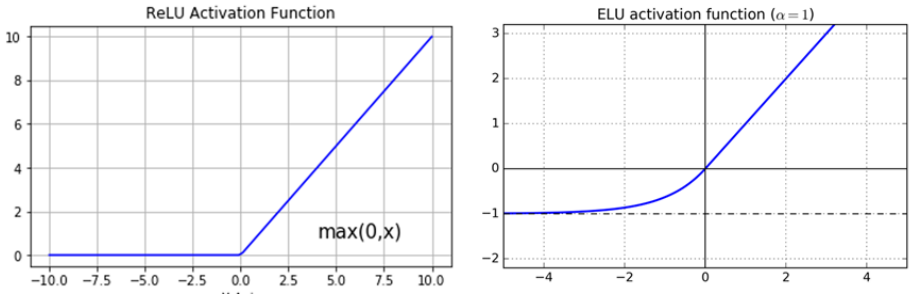

Figure 1: The ReLu and ELU activation functions

2 Literature Review

2.1 EffNet-L2

The current top accuracy for the CIFAR-100 dataset is 96.1%, and was achieved using a EffNet-L2 [3]. The EffNet-L2 simultaneously minimized loss value and loss sharpness and was applied to several different models. However these models themselves were relatively complex and trained from scratch for this, and so were not feasible for implementation in the given time frame of this project.

2.2 Exponential Linear Units

The exponential linear unit (ELU) activation function has been known to get better results than the ReLu activation function when it comes to image classification [1]. This activation function is similar to the ReLu activation function (Figure 1), however it has several distinct advantages, being differentiable and able to return negative values that result in the mean activation being closer to 0, similar to batch normalization. The effect of using ELU as the activation function instead of ReLu is significant, and improves both convergence and the final accuracy.

2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are well known for their image classification abilities, and many of the best models are variations of the basic CNN. Several top CNN models have emerged, and in [10] 3 of the top: AlexNet, GoogLeNet, and ResNet50, were compared on the CIFAR-10 and CIFAR-100 datasets. Overall, AlexNet performed the worst with 44% accuracy on the CIFAR-100 dataset, while GoogLeNet and ResNet50 both achieved similar accuracies of 64% and 60% respectively for CIFAR-100.

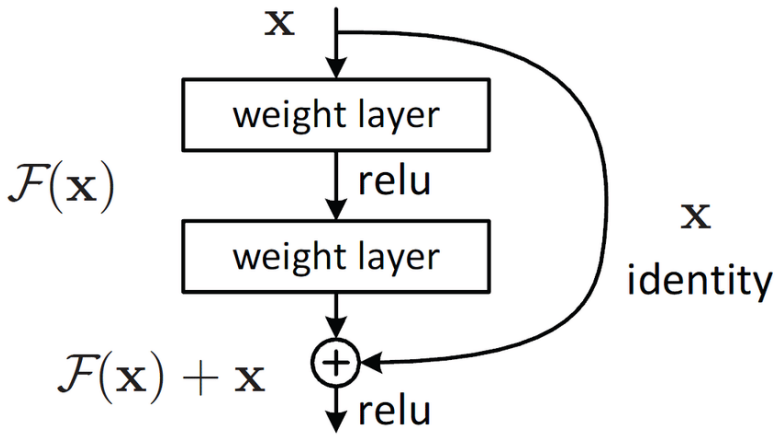

Figure 2: The skip connections used in a residual network

2.4 Residual Networks

Residual networks [5] are one of the best performing models for image classification, more so than the typical CNN. Residual networks are effectively modified CNNs, where instead of each convolutional layer using the output from the previous layer, it uses the output from the previous layer plus the input of the previous layer (Figure 2). This helps to avoid the vanishing gradient and degradation problem common to deep networks, and results in faster convergence.

2.5 Cutout

Data augmentation is often a step needed for image classification, especially when the dataset is limited. Cutout [2] is a type of data augmentation that involves obscuring sections of each image at random in an attempt to improve the robustness and ability to generalize of the model. Overall, this method of data augmentation reported great success, achieving 84.8% on the CIFAR-100 dataset when used with shake-shake regularization models [4].

3 Proposed Solution

CNNs are the primary model used in image classification, so this was used as the base for the models in this project. Research suggested that using pretrained models would be best [7], as the weights would allow for the desired features to be extracted even with another dataset. For this project, two different models were used primarily: the VGG16 [11] model and the ResNet50 [5] model. These models have both shown success when it comes to classifying image data, and are available with Keras. Both models were trained on ImageNet, an incredibly large dataset composed of over 14 million images fitting over 20,000 categories,

though only 1,000 of these were used in the dataset for deep learning models in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

In addition to using pretrained models, when training for the CIFAR-100 dataset several different kinds of data augmentation were used. The basic data augmentation techniques like translation, horizontal flip, and rotation were applied to the training data, along with cutout, where a square section of the image was blacked out. The blacked out square could be centered off the image itself, to allow for some images to have a smaller blacked out section, which is needed to attain a high accuracy.

3.1 VGG16

The VGG16 (Figure 3) is a CNN made up of 16 layers in total. There are 6 blocks of layers, with a max pooling layer in between each block that uses a stride of 2, halving the size of the output with each block. All convolutional kernels are of size 3×3, and the number of filters is the same for all layers in a block. The first 2 blocks have only 2 convolutional layers of 64 and 128 filters respectively, with the next 3 blocks having 3 convolutional layers with 256, 512, and 512 filters respectively. The last block is composed of 3 fully connected layers, the first 2 of size 4096 as hidden layers, and the last of size 1000 as the output layer. This last block of fully connected layers is not included in the pretrained model when it's trained for another dataset however; instead, only the weights of the first 5 blocks are kept. These weights are 'frozen', meaning they won't change when the model is retrained. Fully connected layers with random weights are added on top of these frozen weights, along with the new output layer, and are trained instead. The number of fully connected layers and the number of nodes in them depend on the dataset, and are typically found by trial and error. In the case of this project, 2 fully connected hidden layers of size 1024 were used, with an output layer of size 100. Both hidden layers used the ELU activation function, while the output layer used a softmax activation function.

3.2 ResNet50

The ResNet50 (Figure 4) is a residual network, or a CNN that includes skip or identity connections (Figure 2). Skip connections are when the input of a layer is added to its output, so the next layer's input is a sum of the input it receives from the previous layer and the input sent in to the previous layer. This helps avoid the vanishing gradient and the degradation problem, allowing for the network to become much deeper and to improve its accuracy. The ResNet50 is 50 layers deep, and also composed of blocks of layers. For the ResNet50, it has a single convolutional layer followed by a max pooling layer with a stride of 2. Then it has 16 blocks, all of which have 3 convolutional layers that lead directly into each other, plus a global average pooling and the fully connected output layer with 1000 nodes added to the end. In between each block are the skip

connections, which are simply added to the output of the previous layer as the input and output sizes are the same. The first input layer uses a kernel of size 7×7 with a stride of 2 and 53 filters. After the first layer there are 4 different kinds of blocks that are repeated a different number of times. All blocks use a 1×1 kernel for the first and third layers, and a 3×3 kernel for the middle layer. The first block after the input layer uses 64, 64, and 256 filters for its first, second, and third layers respectively, and this block is repeated 3 times. The second block uses 128, 128, and 512 filters and is repeated 4 times. The third block uses filters of size 256, 256, and 1024, and is repeated 6 times. The fourth and last block before the global average pooling layer uses 512, 512, and 2048 filters, and is repeated 4 times. All these layers in addition to the output layer results in a 50 layer architecture. When using the pretrained model the output layer is replaced with one appropriate for the classification task; in this case, it is replaced with a 100 node fully connected layer. Additionally, a hidden layer with 256 nodes with an ELU activation function was added before the output layer, which used a softmax activation function.

4 Implementation and Validation

The following was implemented in Python 3.8 using the Keras and Tensorflow libraries, version 2.4 for both, and pretrained models.

4.1 Data Processing

The CIFAR-100 (Figure 5) data was acquired from the Keras datasets library, and automatically split into training and testing data. Since there were 60,000 images split evenly into 100 classes, this left 600 images per class. The training set was composed of 500 randomly chosen images per class, with the remaining 100 images saved for the test set. The answers were converted to an $n\times100$ binary matrix as the models were designed for one-hot-encoding.

Data augmentation in the form of affine transformations like translations, rotations, and horizonal flips were applied, and later, so was cutout (Figure 6). The cutout applied was always a square area that set the pixel values to a constant, though the center of the square could be anywhere within the image, including near the edge. This would result in part of the square being off the image, and a smaller blacked out area than normal. Two different sizes were used for the cutout: 5×5 and 8×8.

Since the models were trained on ImageNet, which used images of size 224×224, the CIFAR-100 images had to be upscaled from 32×32 to 256×256. This was so the max pooling layers would be able to reduce the size by half each time, which wouldn't be possible with the original image size.

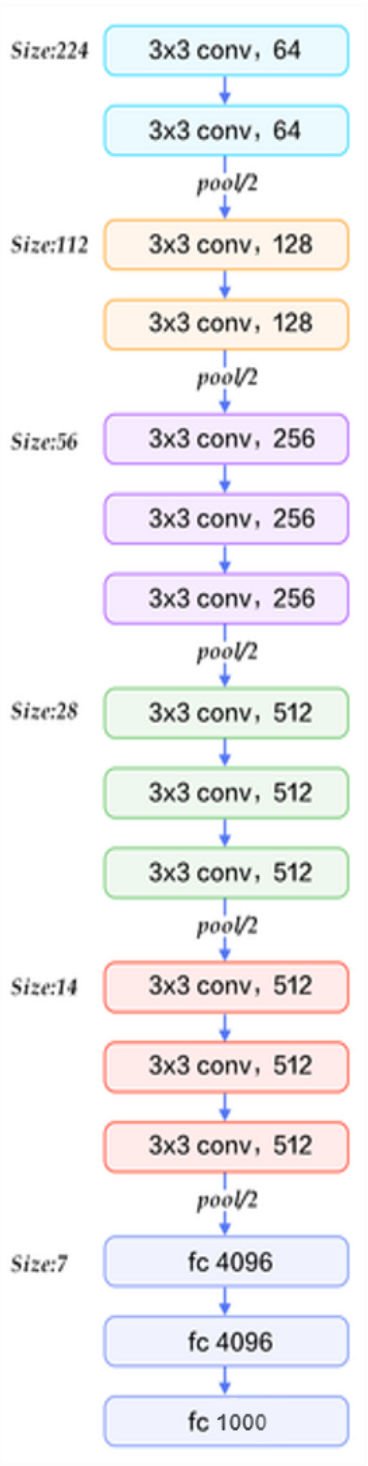

Figure 3: The VGG16 architecture.

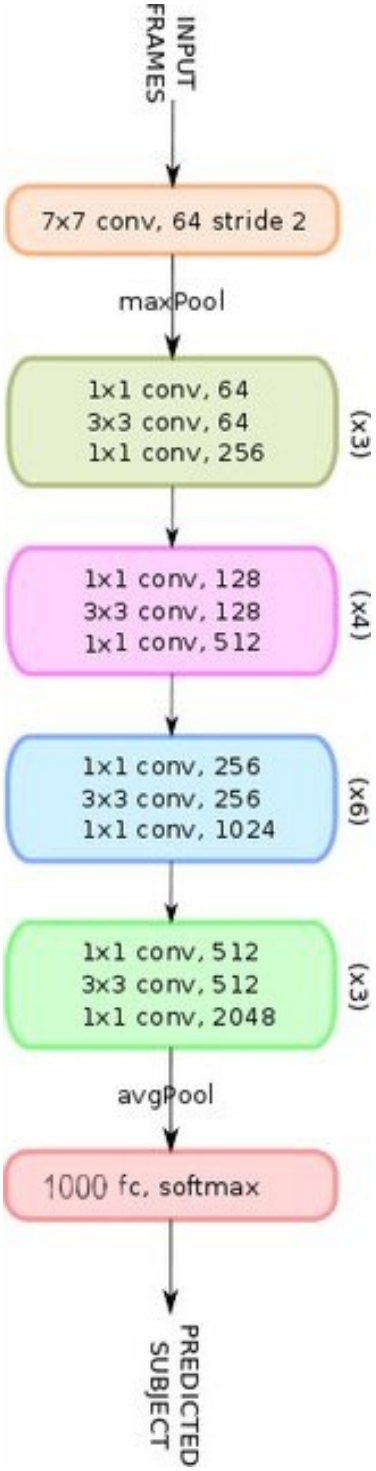

Figure 4: A simplified version of the ResNet50 architecture.

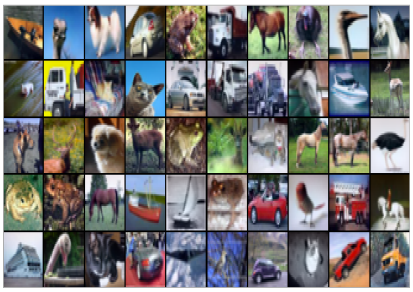

Figure 5: Images from the CIFAR-100 dataset before any preprocessing

Figure 6: Images from the CIFAR-100 dataset with cutout applied

Lastly, each pretrained network has a `preprocess_input` function that was applied to convert the pixel values to the accepted range for the specific model.

4.2 Model Implementation

4.2.1 VGG16

The VGG16 was imported from the Keras library with the ImageNet weights and without the fully connected layers at the end. The output was flattened and fed into 2 fully connected layers of size 1024 that used the ELU activation function. After each of these fully connected layers, a dropout of 20% was applied. The output layer had 100 nodes and used a softmax activation function. A batchsize of 32 was used, and the optimizer was Adam with a default learning rate of 0.001. The loss was calculated using categorical crossentropy, and the model was trained for 5 epochs, as it was observed that this was where the improvement in the accuracy began to plateau.

4.2.2 ResNet

The structure of the ResNet50 model was based on code from https://github.com/balajikulkarni/The-one-with-Deep-Learning/blob/master/TransferLearning/TransferLearning.ipynb, though modified slightly. The data augmentation used included both the built in affine transformations available to Keras and cutout, instead of just cutout. Additionally, cutout was applied to every image instead of just half of them, and was of a fixed size. It could also be centered so that the full size of the cutout was not always entirely on the image, sometimes reducing the size of the cutout area. For the ResNet50, a batch size of 32 was used, and the model trained for 3-8 epochs. The loss was calculated using categorical crossentropy, and accuracy was used as the metric. The Adam optimizer was used, as experimentation had shown it to result in faster convergence. Training for further epochs resulted in no further increase in accuracy for the ResNet50 model when not using cutout. When using cutout,

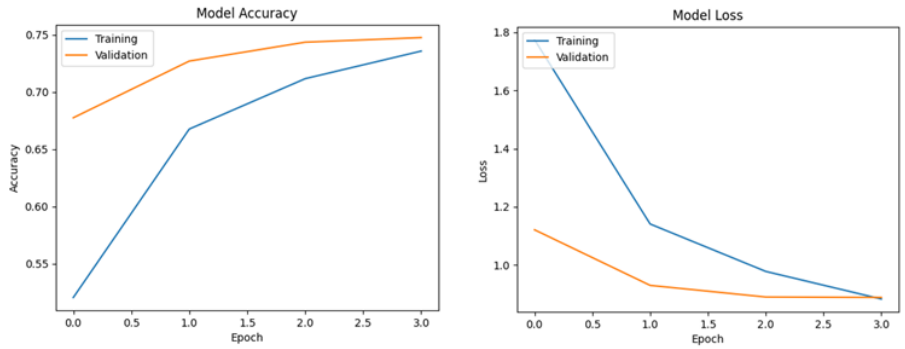

Figure 7: Graphs of the training and validation accuracy and loss for the ResNet50 with only affine transformations used as augmentation.

the training was too slow for a high number of epochs to be reached, and so only went up to 8 epochs, by which point it was unclear as to whether it had plateaued.

4.3 Experimental Environment

The Python code was run on a Windows 10 machine in Pycharm on a single CPU. While a GPU was available, it was incompatible with the code, and running the code on the GPU resulted in the GPU's memory being immediately taken up, and the program crashing.

4.4 Validation Results

The validation metric used was the accuracy, as this was the most relevant and easily understood metric for this project. The parameter values were found by attempting to find trends in the change in accuracy when a value was changed, however this method was rather limited as the number of epochs reasonable to use was too small for extensive testing. It is possible that a much better result would be obtained by using more epochs, however from the trends of the training and validation accuracy and loss, this isn't certain (Figure 7), as even though the training loss seems like it could decrease, the validation loss appears to have plateaued.

Additionally, VGG models were attempted only briefly as they tended to perform poorly, so that model was quickly abandoned. Similarly, using different versions of the same model - ResNet50, ResNet101, etc - did not seem to affect the accuracy significantly, resulting in an improvement of only 2-3%. This made it reasonable to focus primarily on the ResNet50, which performed nearly as well as the larger ResNets, but required less time to train.

Model	Epochs	Affine Augmentation	Cutout Size	Test Accuracy
CNN	15	None	N/A	34%
CNN	15	Yes	N/A	37%
VGG16	5	Yes	N/A	47%
VGG19	5	Yes	N/A	48%
ResNet50	3	None	N/A	75%
ResNet50	3	Yes	N/A	78%
ResNet50	4	Yes	5x5	73%
ResNet50	8	Yes	8x8	76%
ResNet101	4	Yes	N/A	80%
ResNet152	4	Yes	N/A	81%
AlexNet	164	No	N/A	44%
ELU + CNN	25	Yes	N/A	76%
EffNet-L2	200	Yes	10x10	96%

4.5 Critical Discussion

The cutout method did not improve the accuracy as expected, possibly because the implementation was not the same as done in the paper due to the use of the Keras library. In [2] cutout was randomly applied and redone for each epoch, however for my implementation it was randomly applied only once, after which the images were set. This required that the cutout section be significantly smaller than in [2], which had a cutout that took up 64% of the image, being a 32×32 section removed from a 40×40 image. The inability to use a GPU also hampered this project quite a bit, as each epoch for the ResNet50 would take about 3 hours to complete. Training for more epochs likely would have led to a higher accuracy, as the training and validation loss both seemed to be decreasing still by the time it reached its final epoch for the 8×8 cutout.

Another thing to note was that the vast majority of the pretrained models were over trained. For nearly all of the models, the training accuracy was significantly higher than the validation accuracy with the only exception being the models that used cutout. For these models, the training accuracy was approximately the same as the validation accuracy, which suggests that cutout did help to ensure that the models were learning the features. This also suggests that randomly applying cutout at the start of each epoch, or just training for longer, would help improve accuracy even further.

The main thing I learned from this was how to use a pretrained network and apply data augmentation. I also learned that data augmentation is often only applied once, instead of being redone every time, and that it replaces the original image instead of adding to the dataset, which seemed impractical. It is possible to do it every epoch, and to replace the original, but due to time constraints, this wasn't feasible for me to do for this project, though for my research thesis it is likely that I will use this.

5 Conclusion and Future Work

Image classification is still an area undergoing research, and finding models that are better able to generalize is of much interest to many researchers. Any device or project that uses a camera or image data will likely find some use in image classification, and as these become more specialized they will need to classify more specific objects, requiring a model that is able to adapt to new data well. The CIFAR-100 dataset was used to explore how the VGG16 and ResNet50 pre-trained models adapt to new datasets. Overall, the ResNet50 performed better than the VGG16, and augmenting the data further improved performance.

Further augmentation and models could be explored for this dataset, as the current top result achieved (96.1%) for the CIFAR-100 dataset is much higher than the top result (78% for the ResNet50, 81% for the ResNet152) achieved here. As mentioned previously, future work will likely involve actually expanding the training dataset instead of modifying it, and reapplying cutout to the original dataset every epoch. Additionally, training for more epochs would likely help with improving the accuracy too. A possible extension of this project would be to apply object recognition, where a bounding box is drawn around the object it has recognized in the image.

References

[1] Clevert, D. A., Unterthiner, T. & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (ELUs). *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 1–14.

[2] DeVries, T. & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout.

[3] Foret, P., Kleiner, A., Mobahi, H. & Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization.

[4] Gastaldi, X. (2017). Shake-shake regularization.

[5] He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. http://arxiv.org/abs/1512.03385

[6] Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, M. X., Chen, D., Lee, H., Ngiam, J., Le, Q. V., Wu, Y. & Chen, Z. (2019). Gpipe: Efficient training of giant neural networks using pipeline parallelism.

[7] Hussain, M., Bird, J. J. & Faria, D. R. (2018). A study on cnn transfer learning for image classification. *UKCI*.

[8] Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S. & Houlsby, N. (2020). Big transfer (bit): General visual representation learning. *Computer Vision – ECCV 2020 Lecture Notes in Computer Science*, 491–507. https://doi.org/10.1007/978-3-030-58558-7_29

[9] Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

[10] Sharma, N., Jain, V. & Mishra, A. (2018). An Analysis of Convolutional Neural Networks for Image Classification. *Procedia Computer Science*, *132*(Iccids), 377–384. https://doi.org/10.1016/j.procs.2018.05.198

[11] Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.