Residual Attention Networks for Image Classification

Thaminda Edirisooriya
Stanford University
450 Serra Mall, Stanford, CA 94305
tediris@cs.stanford.edu

Abstract

The ImageNet challenge has been the driver for much of the recent advances in Convolutional Neural Network design, with many models such as Inception and ResNet delivering high levels of performance in both the Top 1 and Top 5 tasks. In this work, we explore the performance of architecutres derived from the mentioned works, as well as other, newer design ideas, and evaluate their performance on a small subset of the ImageNet dataset, from here on known as Tiny ImageNet. Specifically, we combine components of the Inception architecture with ideas from ResNet, and from Visual Attention Networks, to achieve reasonable performance on the given dataset.

1. Introduction

The image classification task has been a testing grounds for novel ways of thinking about how machines can make sense of visual information. In recent years, it has pushed the development of a myriad of convolutional neural network designs that have progressively performed better. Following these developments, this work attempts to combine elements from distinct advances in image classification models and create a performant model for the Tiny ImageNet dataset.

The data for this task is a set of 100,000 training input images of size 64x64x3, representing 200 different classes from the original ImageNet dataset. Each class is represented by 500 images, and a total of 100 more images for each class are allocated for test and validation purposes. The desired output of the model is the true class for a given input image, chosen from one of the 200 present in the dataset.

The general architecture of the presented model is as follows: first, we create and apply an attention mask to the input. Next, we feed the result into a modified inception module that decreases the patch size while increasing channel depth. The result is passed into a residual block, and again into multiple inception blocks, after which an average pool and affine transform convert the results into a 200-dimensional vector which generates the output classification.

2. Related Work

Because of the prevalence of work on the image classification task, a lot of influential literature exists, and was heavily drawn upon, when building the model presented here. The literature reviewed falls into three major categories - residual networks, the inceptiom architecture, and visual attention. Components and ideas from each of these developments were used in the final model.

2.1. Inception

One of the major components explored in this work is the use of inception modules as a means of efficiently learning a large set of parameters and learning a set of features for different size convolutions [10]. This work showcased a high-performance method for image classification that was computationally efficient, while also maintaining a light memory footprint. This idea was further explored by the authors of the original paper [11], where different variations of the inception network were developed, such as asymmetric convolutions, convolution decomposition, and other modifications, all in the name of improving performance

Furthermore, recent work on this design has included a discussion of the application of residual networks, detailed in more detail below, on the inception architecture [9]. The authors demonstrate significantly improved training times, and improvements in test accuracy.

2.2. Residual Networks

Also known by the moniker ResNet [2], the addition of residual connections to convolutional neural networks allows for more efficient training of "deep" models - the authors showcase this by comparing the training error for high-depth models with and without the residual connections, and demonstrate a loss curve that drops much faster than a simple, multi-layer convolutional network with identical structure otherwise.

This idea was expanded upon further [3], and different variants of the residual architecture were tested. The work compared the performance of pre vs. post activation for neurons, batch normalization, dropout, gating, and convolution shortcuts, among other experiments [15]. Of the many designs that were tested, it was found that a two layer residual neuron, preceded by a batch normalization operation and ReLU activation, achieved the best results when comparing performance on classifcation on the CIFAR-10 dataset.

Other variations on the residual design include ResNeXt, which incorporates methods similar to the inception architecture of splitting a layer into multiple slices, and applying filters of different sizes to the input [14]. This model attempts tolearn which representations mean the most at a particular layer, eventually combining the results while mainting a residual connection to allow for undiluted gradient backpropagation. Many of the ideas in this work form the inspiration for the model that we present here, as it demonstrates the benefits of combining some of the same building blocks. A similar idea that was not explored was that of stochastic depth, which drops layers at train time to allow gradients to pass through more directly to early layers [4]. While this approach is promising, with the nature of the attention model, it would likely struggle to determine the level at which to choose attention when future dependent layers suddenly went missing.

2.3. Visual Attention

One of the more unique developments in image classification architecture is the use of visual attention to determine patches of an image to focus on, and then classifying based on the area of focus [12]. While attention has been applied very succesfully in the natural language domain [8], the use of attention in visual tasks remains partially unexplored. Elements of this model were used in visual question answering [1], but applying attention in an unsupervised fashion to image classification is a new trend that has only recently been explored, in particular for fine grained image analysis [13]. For example, this specific methodology of using stacked attention layers, in conjunction with residual connections, has been shown to perform well on visual question answering [5].

The specific structure of attention that we explore in this model is the "hourglass" design [6]. It has been shown to work for visual pose estimation, and works by downsampling in input patch before computing an attention mask, then upsampling before applying the final mask.

Finally, a few other interesting models of visual attention have been proposed, such as iterative attention [7].

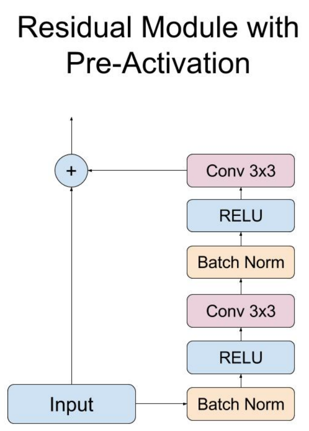

Figure 1: Residual module with multiple preactivations (ReLU), shown to perform slightly better and train faster

3. Methods

The architecture showcased here is composed of multiple layers from modern convolutional neural network literature, assembled together to perform the final classification. The types of modules used in the model are as follows: Residual Preactivated, Residual Inception, Inception Reduction, and Soft Attention. All of these are discussed in detail in the following sections.

3.1. Residual Preactivation

Residual connections between the input to a processing layer, direclty to the output, allow gradients to propogate more easily through a network. They allow for the same expressivity, as the weighted branch can negate the activations that are undesirable. A residual layer can be descirbed as follows:

$$F(x) = x + H(x)$$

where $F(x)$ is the output of the layer, x is the input, and $H(x)$ is the residual function of choice. In this model, the residual component $H(x)$ is defined as two consecutive copies of a trio of operations - batch normalization, ReLU activation, and 3x3 convolution. This whole block $F(x)$ is referred to as a residual model, and is used within other components as a building block.

3.2. Residual Inception

The residual inception module, which draws inspiration from Inception V4[9], allows for a residual computation to flow alongside a standard inception layer. The inception layer is computed as the concatentation of a 1x1 convolution branch, a 3x3 convolution branch, and a double 3x3 convolution branch. The three effective types of convolution filters allow the module to focus on differently scaled

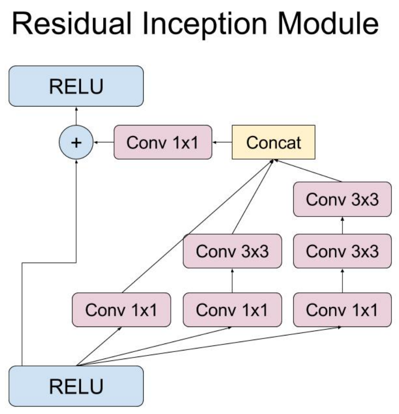

Figure 2: Residual Connections with ReLU activations combined with the inception architecture. Results indicate easier initial learning

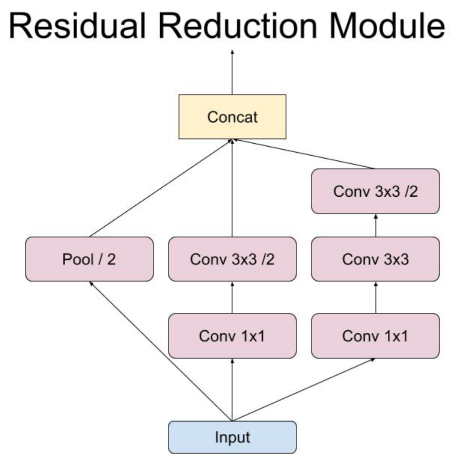

Figure 3: Reduction layer that reduces the dimensionality of the input patch, using techniques from inception to maintain lower level features.

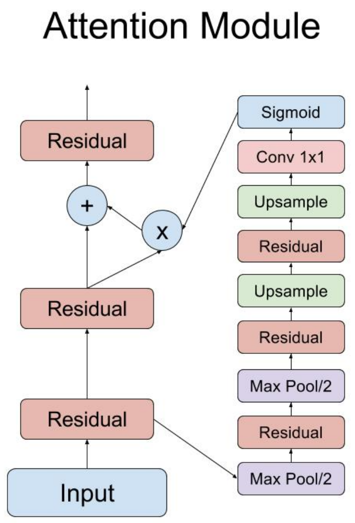

Figure 4: Hourglass Attention Module. Uses the residual blocks defined above

features of interest in the input, and maintain a multi-level understanding of the input patch.

In high dimensional layers, in order to improve efficiency, we downsample the input to the larger branches of the inception module by performing a 1x1 convolution with a smaller number of output filters, and then upsampling the depth channel once we have concatenated the branch outputs.

3.3. Inception Reduction

At multiple stages in the model, we wish to reduce the size of the input patch, while also increasing the depth, with the goal of learning specific features of the input, while also maintaining the depth required to retain enough information to perform classification at the end. Thus, we employ a modified inception module, which uses wider filter strides to reduce the image patch size, while increasing depth.

Specifically, we have a max pooling branch, 3x3 stride 2 branch, and double 3x3 convolution branch with a stride of 2 on the second convolution. Once again, these are concatenated and activated using the ReLU function. We omit the residual connection here for simplicity - including one would require defining a transform from the previous size, diluting the value of the residual connection.

3.4. Soft Attention

The attention implemention used here draws heavily from Residual Attention Network for Image Classification [12]. It consists of two branches - a "trunk" branch $T(x)$ composed of two consecutive residual modules, and an hourglass mask branch $M(x)$[6].

The mask branch operates by downsampling an input patch using average pooling, incorporating a residual module, downsampling again, computing another residual connection, and then finally upsampling to the original size using bilinear interpolation and computing a sigmoid activation. The final output of this module is:

$$F(x) = (1 + M(x)) * T(x)$$

We add 1 to the trunk operation to allow for the trunk branch computation to pass through, where the mask branch works to suppress noise in the trunk branch, allowing for easier detection of important features down the line.

3.5. Final Architecture

The final architecure of our system is thus as follows - a single 3x3 convolution of the input, followed by a Soft Attention layer, the first Inception Reduction layer, a Residual Module, and a second Inception Reduction Layer. Finally, an average pooling operation that is followed by an affine layer, mapping directly to the output classes. It should be noted that I experimented with a third Inception Reduction layer. Diagrams presented here show the final reduction layer, but it should be noted that it is not necessary for the model to perform reasonably.

The overall design of the model is tailored to the task of classifying smaller images - large levels of depth proved difficult for the model to train on effectively, and it would either overfit the training data, or fail to build a reasonable feature mapping for the images and resulted in low accuracy. Conversely, less expressive models that were attempted did not have the complexity to model the necessary feature extraction that the task of ImageNet classification requires.

The final output loss is computed as the softmax cross entropy of the predictions generated by the affine layer. The softmax of a vector is defined as follows:

$$S(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

where $S(x)$ is the softmax function, and x is an input vector. The loss is defined as:

$$L(x) = -\sum_i \log(S(x)_i) y_i$$

so we encourage maximizing the score of the correct class, while minimizing scores of the incorrect classes.

4. Dataset and Features

As mentioned earlier, the Tiny ImageNet dataset contains 100,000 training images, 10,000 validation images, and 10,000 test images, all evenly distributed among the 200 classes selected for this dataset. All images are 64x64x3 with the 3 channels representing the red, green, and blue channels of the input images. During the preprocessing step, we subtract the mean of the test images so that they are centered. We omit normalizing the magnitudes using variance due to the early application of a batch normalization operation.

5. Results and Discussion

Throughout the development of the final model, 3 architectural variations were tested - one without the attention mask and the last inception reduction, one without the last inception reduction, and the full model as described above.

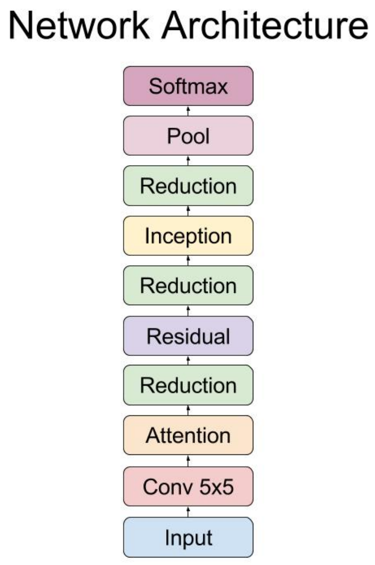

Figure 5: Full architecture of the largest model. Experiments were done with and without the final reduction layer, as well as substitutions for the attention layer

Figure 6: Sample images from the dataset

We evaluate model performance by comparing validation accuracy for the trained models.

5.1. Training Details

The Adam optimization method was used to minimize the softmax cross entropy loss discussed earlier. Training

Layer	Size
Conv 3x3	64 x 64 x 64
Soft Attn	64 x 64 x 64
Reduction	31 x 31 x 128
Residual	31 x 31 x 128
Reduction	15 x 15 x 320
Inception Residual	15 x 15 x 320
Reduction	7 x 7 x 832
Avg Pool	1 x 1 x 832
Affine	200

Figure 7: Layer specific paramters for the whole architecture

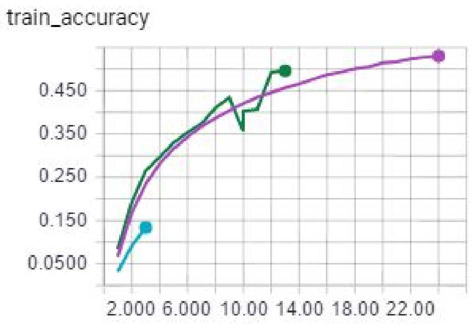

Figure 8: Training Accuracy for 3 of the models. Purple: Attn-Simple, Blue: Attn-Reduction, Green: No-Attn Extra

was performed in mini-batches of size 32, to allow the GPU memory of the training machines to store all the parameters and images necessary for a batch computation. Validation accuracy and loss was assessed at the end of each training epoch, on the whole validation set. Learning rate started at 0.001, and was decayed at a rate of 0.96 every 1000 iterations. When learning rate was not allowed to decay, we observe that the model gets "stuck" at certain loss values, and upon manually setting the learning rate mid-training, that the model once again picks up and continues to improve. Training data was randomly shuffled for each epoch, and statistics were aggregated over each epoch for visualization purposes.

5.2. Hyperparameters

Dropout was applied to the output of the final average pool operation, before the classification stage of the architecture. A dropout rate of 0.4 was chosen after a small hyperparameter search on one of the first models, and it continued to show the desired effect on the later models. Overfitting was only observed many iterations after the validation loss had converged, and validation loss and accuracy never worsened during training, even in later iterations.

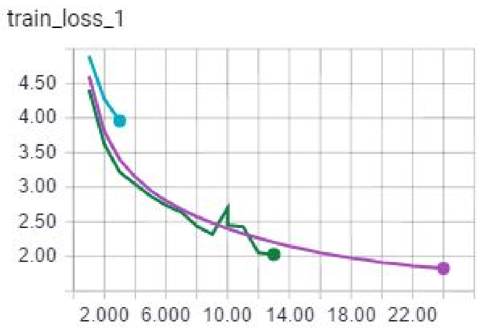

Figure 9: Training Loss for 3 of the models. Purple: Attn-Simple, Blue: Attn-Reduction, Green: No-Attn Extra

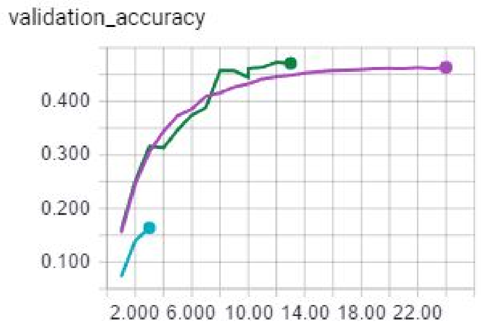

Figure 10: Validation Accuracy for 3 of the models. Purple: Attn-Simple, Blue: Attn-Reduction, Green: No-Attn Extra

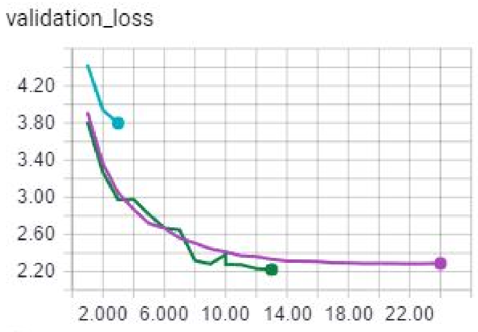

Figure 11: Validation Loss for 3 of the models. Purple: Attn-Simple, Blue: Attn-Reduction, Green: No-Attn Extra

5.3. Inception Reduction Performance

Filter depth at specific layers was chosen using a combination of design decisions made in the references, and experimentation. For the different architectures tested, performance was evaluated as a function of validation loss at each iteration, as well as the loss/accuracy curves. One of the standout results was that adding the final reduction stage prior to the classification step had a detrimental effect on the ability of the model to train properly, even with all of

Model	Val Acc (%)
No Attention	48.0
Attn-Simple	46.7
Attn-Reduction	32.7
No Attention Extra	48.2

Figure 12

the residual connections. Train loss would continue to decrease, but validation loss would stop improving after a few epochs, even with high dropout values.

5.4. Other Experiments

Another experiment that was attempted was using a two-layer affine network to perform the classification - the intuition behind this being that a more complex function of the generated features may be necessary to perform classifiction. Results from early training iterations showed very similar performance, and while training loss improved marginally faster, validation scores were very similar early on. However, by modifying the dropout probabilities of the classification layer inputs as training progressed, larger improvements in the results were evident.

The results from the improved classification layer on the No-Attention model suggest that, when used in combination with the attention module, could allow the increasd expressivity of the attention model to manifest in the improved performance that we would expect for having increased model complexity.

6. Conclusion

From the results gathered, the high-level architecture appears to be fairly performant. While the performance numbers of simplest model appear to be the best, this is very likely due to the lack of training time - models were run for at most 25 epochs, and many of the more complex models for less than that. One trend that does seem to hold, though, is that on the Tiny ImageNet dataset, highly complex and deep models do not perform as well as one might hope, due to the relatively small size of the training data. When working with only 500 examples of each class, it becomes very easy for complex models to overfit, which is demonstrated by the poor performance of the full attention with 3 reductions model.

The work was heavly limited by computation time for experimenting with different models, and since the architecture is an attempt at a novel combination of other works, there was no pre-trained data to serve as a starting point for the model. It is likely that the initial convolution layer is not necessarily extracting the features we care about as well as we would hope, but there was not enough time to test different initial convolution sizes, different reduction patterns, and other smaller archicture choices.

There are some promising directions to explore with this work - using different residual modules, such as ResNeXt[14], could very likely improve performance. Experiments on different architectures for the classification layer is also something that could lead to better results - one of the final experiments explored this idea, but the experiment was not run for long enough, or with as much hyperparameter exploration as may have been necessary to induce good performance.

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[4] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016.

[5] J. Kim, S. Lee, D. Kwak, M. Heo, J. Kim, J. Ha, and B. Zhang. Multimodal residual learning for visual QA. *CoRR*, abs/1606.01455, 2016.

[6] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016.

[7] M. F. Stollenga, J. Masci, F. J. Gomez, and J. Schmidhuber. Deep networks with internal selective attention through feedback connections. *CoRR*, abs/1407.3068, 2014.

[8] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[9] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[12] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. *CoRR*, abs/1704.06904, 2017.

[13] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. *CoRR*, abs/1411.6447, 2014.

[14] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.

[15] S. Zagoruyko and N. Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.