A Review of Image Classification

Marco Wong
Department of Computer Science
University of Toronto
Scarborough, ON M1C 1A4
`marcoo.wong@mail.utoronto.ca`

Chui Ling Mok
Department of Computer Science
University of Toronto
Scarborough, ON M1C 1A4
`chui.mok@mail.utoronto.ca`

Abstract

Convolution neural networks have shown significant performance on image classification tasks. The most notable models during the early 2010s were VGG and ResNet with 19 and 152 layers, respectively. CNNs had learned to represent image features through combinations of convolutional layers and max pooling with fully connection layers for classification. We plan to evaluate pre-trained models of VGG and ResNet provided by torchvision to classify sets of images and investigate the discrepancy in classification probabilities.

1 Introduction

Over the years, the original ResNet-34 and [VGG stuff] were modified for accuracy improvement and task specific limitations. We have witnessed change in depth, width and the addition of skip connections for computational limits and data set complexity. Our goal is to reproduce experimental results while tuning hyperparamters and model selection to maximize validation accuracy. We first analyze model variations of ResNet and VGG, then provide insights to why some variations perform better and why ResNet outperforms VGG.

2 Related Works

ResNet and VGG are both used for image classification but differ in some aspects. It's simpler to implement VGG as its architecture is fairly straight forward and has fewer hyperparamters. For tasks such as video classification, ResNet performs better because of its residual connections and scalability. ResNet had it's history of model improvements (Stem modification He et al. [2018], downsampling blocks, etc.), but few researchers had scaling propositions.

3 ResNet-RS

We will be reimplementing the suggested changes [Bello et al. [2021]] to ResNet in attempt to show training and scaling strategies affect performance more than network changes. We employ and review the following changes; (1) model depth is scaled for cases of overfitting (2) image resolution is increased slower than recommended [Tan and Le [2020]]. The redesigned ResNet model was called ResNet-RS, which was stated to perform $1.7x - 2.7x$ times faster than EfficientNets on Pareto Curve. Performance of the new ResNet could also be extended to other tasks such as video classification. The figures and results were compiled from Bello et al. [2021]. The GitHub repository to the code of ResNet-RS, created by Nachiket. [1]

[1] `https://github.com/nachiket273/pytorch_resnet_rs`

3.1 ResNet-RS Architecture

The ResNet-RS model was an architecture composed of ResNet-D (7.1) and Squeeze-and-Excitation (7.2). Figure 1 describes the composition of ResNet-RS. The model accepts 224x224 image resolution The $x3$ in the convolutional layer indicates the block is repeated three times. The block weights would change depending on the ResNet depth (ResNet-50 vs ResNet-101). The corresponding blocks weights can be found in 7.3. Some things to note, adding blocks in the lower layers networks (ResNet-50) reduces overfitting because those networks have fewer parameters.

Block Group	Output Size	Convolution Layout	
stem	112x112	3x3, 64, s2 3x3, 64 3x3, 64	x1
c2	56x56	1x1, 64 3x3, 64 1x1, 256	x3
c3	28x28	1x1, 128 3x3, 128 1x1, 512	x4
c4	14x14	1x1, 256 3x3, 256 1x1, 1024	x23
c5	7x7	1x1, 512 3x3, 512 1x1, 2048	x3
	1x1	Avg Pool Dropout 1000-d FC	x1

Figure 1: ResNet-RS

3.2 ResNet-RS Training

To further prevent overfitting, techniques such as weight decay, smoothing, dropout, and stochastic depth were implemented for regularization. Stochastic depth Huang et al. [2016] is similar to dropout, instead it drops layers or blocks within a probability as a function of depth. Data augmentation was another regularization technique used It applied a series of random image transformations (crops, flips, noise) to image data. For hyperparameter tuning, a held-out validation set with 2% of ImageNet training set was used.

3.3 ResNet-RS Results

In Figure 2, each colour represents the additional method on top of the baseline ResNet-200. Top-1 column represents the performance metric of the model on ImageNet accuracy set. For example, the ResNet-200 with Cosine LR Decay has 79.3% top-1 accuracy at a 0.3% increase from the baseline, ResNet-200. The colours describe the change: Purple(Training Methods), Green(Regularization Methods), Yellow(Architecture Changes) The negative delta when increasing training epochs is the result of over-fitting with more epochs. Increasing epochs must be used in conjunction with other techniques for its benefits. Furthermore, the negative delta caused by Dropout on FC would be fixed once we decrease weight decay.

Improvements	Top-1	Δ
ResNet-200	79.0	—
+ Cosine LR Decay	79.3	**+0.3**
+ Increase training epochs	78.8 [†]	-0.5
+ EMA of weights	79.1	**+0.3**
+ Label Smoothing	80.4	**+1.3**
+ Stochastic Depth	80.6	**+0.2**
+ RandAugment	81.0	**+0.4**
+ Dropout on FC	80.7 [‡]	-0.3
+ Decrease weight decay	82.2	**+1.5**
+ Squeeze-and-Excitation	82.9	**+0.7**
+ ResNet-D	83.4	**+0.5**

Figure 2: Accuracy Improvement

3.4 ResNet-RS Significance and Drawbacks

The results of ResNet-RS showed significant accuracy improvement (+3.2%) from training and regularization methods alone. However, training methods are sometimes task-specific and do not always generalize well. Additionally, by adding two fairly simple architecture changes, we witnessed accuracy increase by +1.2%. But architecture improvements are not guaranteed to perform at larger image resolutions. The author found scaling to depth works better for large epochs whereas scaling to width works well with fewer epochs. Scaling techniques aren't certain to work high pixel densities. While architecture changes show improvement, simplicity may be preferred for speed and computation availability. Even though ResNet was introduced in 2015, it still shows exceptional strength with modern changes.

3.5 ResNet-RS Applications

The same training and architecture changes were applied to another model, 3D ResNet-50 to create an improved 3D ResNet-RS model. It was meant for video classification on Kinetics-400 which is dataset containing 240,000 video clips from 400 categories with an average length of 10 seconds. Slight modifications to the changes were made; removed Cosine LR decay and added Scale jittering. Scale jittering creates new frames by scaling within a predefined factor. The authors achieved 78.2% Top-1 accuracy at a +4.8% increase from the baseline 3D ResNet-50.

4 VGGNet

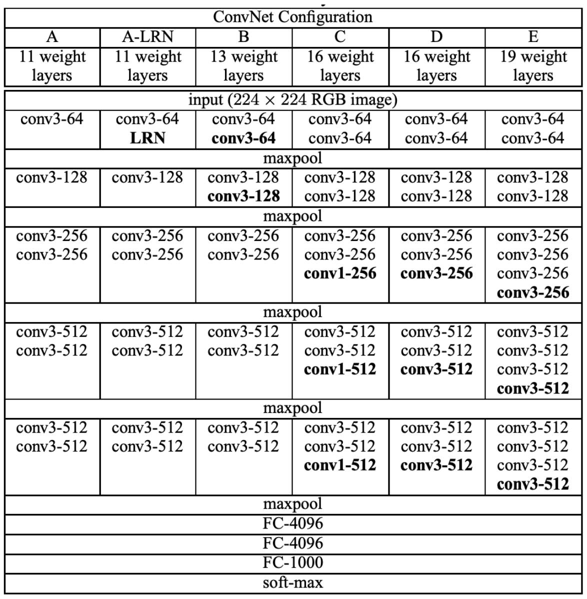
VGGNet (Simonyan and Zisserman [2015]) is a deep CNN model developed by the Visual Geometry Group (VGG) at the University of Oxford. Researchers at VGG hypothesized that increasing the depth of the network could help the model learn more complex and abstract features from images. VGG19 was designed to be deeper than existing CNN architectures at that time, such as AlexNet and GoogLeNet. The model has been widely used in various computer vision tasks, such as image classification, object detection, and image generation. There is a GitHub repository to the code of VGGNet. [2]

4.1 VGGNet Architecture

Figure 3 shows the structure of different variations of the VGGNet model. The input layer takes RGB images of size 224x224x3, and each convolutional layer is followed by a ReLU activation function. Max pooling layers with a pool size of 2x2 and a stride of 2x2 are used to reduce spatial dimensions while dropout layers are used for mitigating overfitting.

4.2 VGGNet Training

VGGNet is trained using a similar method used by Krizhevsky and Hinton [2012] using stochastic gradient descent (SGD) with mini-batch updates. During training, the authors used a method called scale jittering (7.4) which involves randomly resizing the training images to different scales. The model is initialized with small random weights and is fine-tuned using backpropagation. Regularization techniques, such as weight decay and dropout, are used to prevent overfitting. Weight decay adds a penalty term to the loss function that discourages large weights, while dropout randomly drops out a fraction of the units during training to prevent the model from relying too heavily on any single unit.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3: VGG19 structure. Image from

4.3 VGGNet Results

Simonyan and Zisserman [2015] performed multi-scale evaluation by resizing test images to different scales and passing them through the trained VGGNet models to obtain predictions. Figure 3 presents the top-1 and top-5 classification error rates of the VGGNet models at different scales.

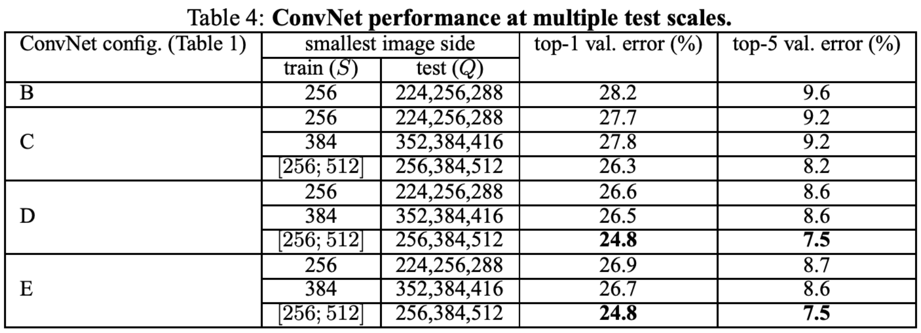


Table 4: **ConvNet performance at multiple test scales.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

Figure 4: VGG results. Image from

[2]`https://github.com/pytorch/vision/blob/main/torchvision/models/vgg.py`

The results suggest that VGGNet performs better at larger scales, indicating their ability to capture objects of different sizes. Scale jittering helped the model to learn to recognize objects at different scales and improve its ability to handle images with varying object scales during testing. Some drawbacks of the VGGNet model includes having relatively large number of parameters, high computational cost, limited efficiency and overfitting potential.

4.4 VGGNet Application

Other than image classification, VGGNet is used in a wide variety of computer vision applications such as medical image analysis, autonomous vehicles and video analysis applications. The model was also used in numerous research papers as a basis for further advancements. For example, Christian Szegedy and Rabinovich [2015] introduced the Inception architecture while using VGGNet as the baseline, and Kaiming He and Sun [2014] proposed the Spatial Pyramid Pooling (SPP) layer to improve the accuracy of the VGGNet.

5 ResNet vs. VGG

5.1 Architecture

VGGNet is a series of CNN architectures with different depths consisting of only convolutional and fully connected layers with a fixed number of layers. On the other hand, ResNet is a CNN architecture that introduced the concept of residual connections, allowing for much deeper networks with skip connections that bypass certain layers. ResNet's deeper architecture allows it to learn more complex and abstract features from images, potentially leading to improved accuracy.

ResNet introduced skip connections, where the input to a certain layer is added to the output of a subsequent layer. This allows ResNet to mitigate the vanishing gradient problem and enables effective training of very deep networks. VGGNet, on the other hand, does not have skip connections and relies on stacking multiple convolutional layers to learn complex features.

5.2 Computational efficiency

ResNet's skip connections allow for better gradient flow and enable the training of deeper networks. This results in improved computational efficiency as compared to VGGNet, which can suffer from vanishing gradients in deeper layers. ResNet achieves similar or better accuracy than VGGNet with fewer parameters, making it more computationally efficient.

5.3 Performance

ResNet has shown better performance than VGGNet in many computer vision tasks, including image classification, object detection, and image segmentation. ResNet has won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) multiple times, with an accuracy of 3.57% compared to VGGNet's 6.67% in 2014. VGGNet tends to suffer from overfitting.

5.4 Complexity

VGGNet has a simpler architecture compared to ResNet, with only convolutional and fully connected layers. ResNet, with its skip connections and residual blocks, can be more complex and may require more computational resources for training and inference.

6 Conclusion

In summary, while VGGNet and ResNet are both popular CNN architectures, ResNet's introduction of skip connections and deeper architecture has shown improved performance and computational efficiency compared to VGGNet in many computer vision tasks. ResNet's flexibility and scalability also make it a preferred choice for many deep learning applications. However, the choice between VGGNet and ResNet ultimately depends on the specific requirements of the task at hand, available computational resources, and the trade-offs between model complexity and performance.

7 Appendix

7.1 ResNet-D Architecture

ResNet-D combines the following four adjustments to the original ResNet architecture. First, the 7×7 convolution in the stem is replaced by three smaller 3×3 convolutions, as first proposed in Inception-V3 (Szegedy et al., 2016). Second, the stride sizes are switched for the first two convolutions in the residual path of the downsampling blocks. Third, the stride-2 1×1 convolution in the skip connection path of the downsampling blocks is replaced by stride-2 2×2 average pooling and then a non-strided 1×1 convolution. Fourth, the stride-2 3×3 max pool layer is removed and the downsampling occurs in the first 3×3 convolution in the next bottleneck block.

7.2 Squeeze-and-Excitation Architecture

Squeeze-and-Excitation reweighs channels via cross-channel interactions by average pooling signals from the entire feature map. For all experiments we use a Squeeze-and-Excitation ratio of 0.25 based on preliminary experiments. In our experiments, we sometimes use the original ResNet implementation without SE (referred to as ResNet) to compare different training methods. Clear denotations are made in table captions when this is the case.

7.3 ResNet Block Weights

Model	Depth	Block Configuration
ResNet	50	[3-4-6-3]
ResNet	101	[3-4-23-3]
ResNet	152	[3-8-36-3]
ResNet	200	[3-24-36-3]
ResNet	270	[4-29-53-4]
ResNet	350	[4-36-72-4]
ResNet	420	[4-44-87-4]

7.4 Scale Jittering

This can be achieved by resampling the images at different resolutions, either by downsampling or upsampling the images, while maintaining their original aspect ratio. The resized images are then used as additional training data for the model.

References

Irwan Bello, William Fedus, Xianzhi Du, Ekin D. Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies, 2021.

Yangqing Jia Pierre Sermanet Scott Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Christian Szegedy, Wei Liu and Andrew Rabinovich. Going deeper with convolutions, 2015.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks, 2018.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth, 2016.

Shaoqing Ren Kaiming He, Xiangyu Zhang and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition, 2014.

Sutskever I. Krizhevsky, A. and G. E. Hinton. Imagenet classification with deep convolutional neural networks, 2012.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.