Image Classification with ImageNette

Caspar Chan (z5206252), Fenyi Shen (z5323706), Henry Ho (z5312521), Rachael Yu (z5164297), William Wong (z5205428)

I. Introduction

Image classification is a popular problem that has been investigated by numerous researchers with several existing solutions generalised for different types of datasets. In this project, we investigated some of those popular model architectures, namely MobileNetV2 and Resnet as the baseline of our own solution and made improvements upon them targeting the ImageNette dataset. For the experiments, we aimed to improve the accuracy of the model with different conventional approaches on model construction and training. Apart from improving the accuracy, we also aimed to evaluate the effectiveness of those approaches on our specific models.

II. Methods

After researching popular neural networks used for image classification, we began by replicating GoogLeNet, MobileNet and various versions of ResNet (ResNet34, ResNet50, ResNet50-d as proposed by (He T. , et al., 2018). Our experiment showed that ResNet models delivered a higher accuracy than GoogLeNet for the ImageNette dataset. We then attempted to boost the accuracy of ResNet through various aspects such as modifying the internal architecture, replacing specific features such as activation function and pooling methods, adding dropout layers, and incorporating learning rate schedulers into the training cycle. Our final model is a composition of various adjustments that experiments had shown to be successful at increasing the test accuracy on the ImageNette dataset.

ResNet bottlenecks allow information to be preserved as it passes through the neural network by using shortcut connections that connect the identity of each bottleneck and appends it to the result of a set off convolutions. Our model builds upon this by merging 2 divergent sets of convolutions and the identity into a combined output under the assumption that performing 2 passes on the input would draw out more information compared to using either by themselves. However, doing so doubles the computation power and space required for each block, making it harder to scale. In addition, there must be enough contrast within the blocks for the network to avoid overemphasizing the weights over the identity. Hence, we chose to use an inverted residual block from MobileNetV2 to pair with our standard residual block as it performs similarly to a standard residual block but is also more memory efficient.

Our implementation of the neural network builds upon this combined bottleneck and follows MobileNetV2's architecture as its main structure. To simplify our model, we use the same number of intermediate channels for both passes and perform down sampling at the start of every block. We used an expansion rate of 6 for our intermediate channels within the blocks as recommended by (Sandler, Howard, Zhu, Zhmoginov, & Chen). Dropout was not included as experimental results did not demonstrate any major improvement in training accuracy while drastically increasing the training time and including it in all bottlenecks causes our network to fail to learn. Furthermore, after every convolution, we employ batch normalization and replaced ReLU activations with Mish as proposed by (Misra & Landskape, 2019) which improved training efficiency. The use of a learning rate scheduler greatly reduced the number of epochs required to reach a test accuracy of 80%, upon comparison we settled with the cosine annealing scheduler.

III. Experimental Setup

The dataset consists of ten classes of images of various things such as dogs and trucks, etc. Each class has a similar number of samples ranging from 858 to 993. Images have differing sizes although they are all relatively small (typically between 200 and 500 pixels in dimension). This dataset includes premade training and validation sets, with 9469 training samples and 3925 validation samples. Also worth noting is that some images (~5%) are incorrectly labelled which makes it more relevant to realistic applications as data quality is rarely perfect. This leads to better generalisation and robustness of the model.

Transformations were done on both the training and validation sets as part of image pre-processing to improve model training.

Resize resizes the images to the size specified. Size 280 was chosen because it performed best amongst a range of other sizes between 150 and 450. RandomCrop crops the images at random points. This helps the model to generalise better as objects could be in various positions or partially obscured in the images. Size 224 was chosen because the amount of cropping is sufficient but still large enough so that important features remain. RandomHorizontalFlip flips the images horizontally at random. This also helps the model to generalise better as objects could be in different orientations in the images. Normalise reduces numerical values in images to a common scale. Different values were tested to find an optimal mean and standard deviation. RandomErasing erases a random rectangular region in images. This works in a similar fashion to RandomCrop in that it helps the model generalise better by obscuring parts of objects in the images.

The validation set has all the same transformations except for RandomHorizontalFlip and RandomErasing, because the purpose of the validation set is to validate the success of the model on realistic data.

For our main loop, to calculate loss we used Cross Entropy Loss and for our optimizer we found that using Adam with an initial learning rate of 0.00024 and weight decay of 0.00001 outperforms SGD as recommended by (He K. , Zhang, Ren, & Sun, 2015) since it allows our model to initially converge faster. We used a mini-batch size of 32 as experimentation between 32, 64, 96, and 128 shows that while the time taken to train each epoch is inversely proportional to the batch size, it also causes our model to overfit earlier. Tests were also conducted to see if applying a learning rate scheduler improved our accuracy and we settled on using a cosine annealing schedule with T_max set to 10.

IV. Results

1. Merging features from different models can bring more insights on new model architecture

Based on the experimental results in Table 1. It is found that the modified MobileNetV2 generally outperforms the ResNet50 except for two classes. This implies that combining the Resent residual block with the MobileNetV2 residual block slightly improves accuracy. As we did not use any pre-trained models, the outcome is believed to be unbiased.

2. The size and purity of the training dataset can have an impact on the accuracy

As shown in Table 1, the Raw Resnet50 only achieves an 87 % accuracy. However, according to an online database (Meta AI), Resnet50 should have the ability to achieve an accuracy above 95% for the ImageNet dataset. The underperforming reason is suspected to be the insufficient dataset size, as the dataset we used is only a subset of ImageNet. Also, it is found that the training dataset contains about 5% noisy data, which is considered as a major factor in lower accuracy. The presence of the noisy data increases the model complexity and time of learning which degrades the performance of learning algorithms.

3. Dropout layers do not necessarily improve the model performance

Although it is mentioned in numerous pieces of literature that dropout regularization is a generic approach to improve the accuracy by reducing overfitting features. However, it is not the case in our experiment.

It is suspected that batch normalization in our implementation already has a regularizing effect (Ioffe & Szegedy, 2015). Literature also suggests that partial dropout can be harmful in CNN because the spatial relationships encoded in feature maps and activations can be highly correlated, (Jansma, 2022). The comparison of dropouts with different probabilities and batch normalization is shown in Figure 1.

4. Fine-tuning hyperparameters is essential to achieve a high accuracy

During our experiment, we tried to run ResNet50 with a consistent 0.001 learning rate. It is found that the accuracy could only barely reach 80%. However, after we implemented the CosineAnnealingLR scheduler, the accuracy went up to 87%. This outcome matches the results from another research study (Peng & Wang, 2012).

5. Mish activation improved training speed and BlurPool caused a decrease in model's accuracy

During the experiment, we first tried training with reLU6 activation. The accuracy goes rather high at 85%. Replacing it with Mish activation do not have any improvement in accuracy, but improved training speed. The Mish activation is faster than reLU6 in reaching 80%.

BlurPool applies a spatial low-pass filter before pooling operations, it is supposed to increase train accuracy without slowing down the learning process. Our experiment showed the opposite, probably the result of degradation. (Zhang, 2019) suggested the replacement of maximum pooling, stride convolution, and average pool with various sequences involving the use of blur pool offered increasing accuracy and better generalisation in commonly used architectures for the ImageNet dataset. However, our experiments showed instead a decrease in accuracy when the blurpool is applied.

V. Conclusions

In this project, we found that merging features from different models can bring insights on new model architecture. We have tried different techniques and approaches in model construction and training. We confirmed that hyper-parameter tuning is essential regardless of the model complexity. We also found that mish activation would improve the training speed while blurpool have an adverse effect on accuracy. Also, dropout layer does not necessarily improve the model accuracy.

However, due to the lack of computation power (GPU), we are unable to conduct more trials to confirm our results. Since each modification takes a long time to obtain the outcome, this adds towards the strict time constraint. As a result, with better hardware and more time, we believe that more in-depth experiments and more sophisticated results can be generated.

Bibliography1

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition.* Microsoft Research.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2018). Bag of Tricks for Image Classification with Convolutional Neural Networks. *Arxiv*.

Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* Google Inc.

Jansma, H. (2022, August 23). *Don't Use Dropout in Convolutional Networks*. Retrieved from KDnuggets: https://www.kdnuggets.com/2018/09/dropout-convolutional-networks.html

Meta AI. (n.d.). *Image Classification on ImageNet.* Retrieved from https://paperswithcode.com/sota/image-classification-on-imagenet?metric=Top%205%20Accuracy&tag_filter=3

Misra, D., & Landskape. (2019). Mish: A Self Regularized Non-Monotonic Activation Function. *Arxiv*.

Peng, P., & Wang, J. (2012). *How to fine-tune deep neural networks in few-shot learning?* Hangzhou: Zhejiang University.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (n.d.). *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* Google Inc.

Zhang, R. (2019). Making Convolutional Networks Shift-Invariant Again. *Arxiv*.

Appendix

Table _: MobileNetv2 and Our Model's Architecture

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Table 1: Model accuracy between Modified MobileNetV2 and Resnet50 on different classes

Class Accuracy	Modified MobileNetV2	Raw Resnet50
Tench	93.5 %	93.0 %
English Springer	93.4 %	95.9 %
Cassette Player	89.9 %	81.2 %
Chain Saw	79.5 %	74.4 %
Church	91.7 %	90.2 %
French Horn	86.3 %	91.9 %
Garbage Truck	91.3 %	89.5 %
Gas Pump	80.4 %	77.8 %
Golf Ball	90.0 %	84.7 %
Parachute	91.0 %	89.7 %
Overall	89 %	87%

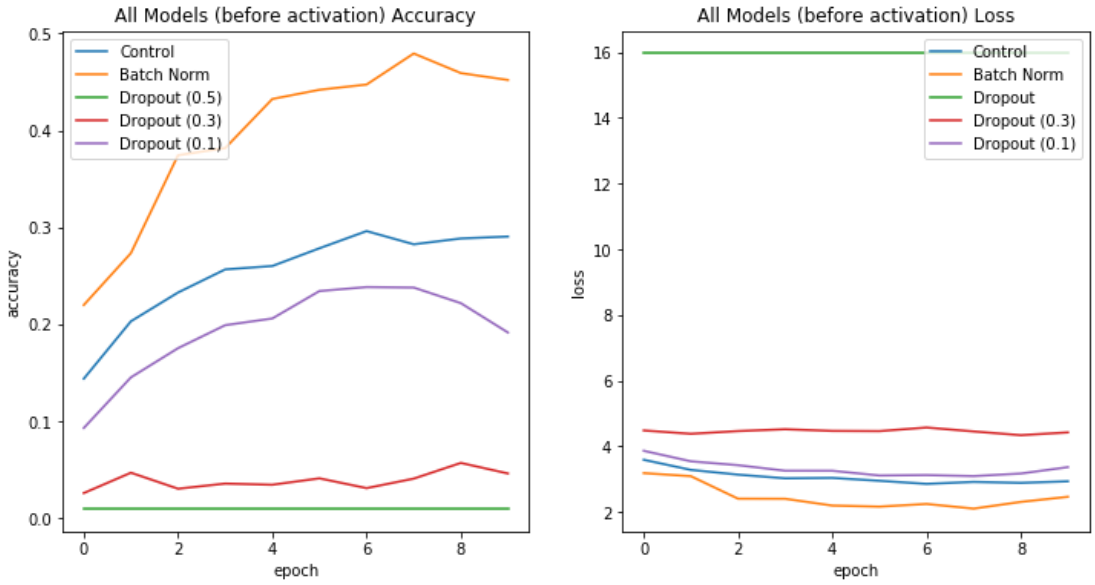

Figure 1: Dropout vs Batch Normalization in terms of accuracy and loss