

Convolutional Neural Networks for Image Classifications

Final Report

Group members:

Yu Yun	3035771701
Zhang Huixi	3035772133
Fung HoKit	3035779105
Jing Jingderong	3035771775
Mi Hongyu	3035756385

researchers and engineers from various colleges and companies.

1. Introduction

Convolutional Neural Network(CNN), one of the most widely adopted models in the field of Supervised Learning, has proved to be highly efficient and accurate when solving image recognition problems in practice. Wondering about the fundamental rationales of such a fascinating approach, our team attempts to solve the image recognition problem described below using CNN models while exploring various architectures and characteristics of CNN. Furthermore, we attempted to optimize our solution with different datasets, optimizers, as well as parameter tuning combinations. Finally, a relatively ideal model is obtained: ResNet50 trained by CIFAR-10, with learning rate equal to 10^{-2} and weight decay is 10^{-3} , along with the Adadelta optimizer,

2. Data processing

Our group aims to address place recognition with the features automatically learned from data. First, we propose a graph-based visual place recognition method. The graph is constructed by combining the visual features extracted from convolutional neural networks (CNNs). The CNN was formed based on a large dataset that contains 17 kinds of flowers with 80 images in each. We plan to split the above images in each class into three subsets (training, validation, and test set). The training set and the validation set will be used for model training, while the test set will be for performance evaluation.

3. Proposed Methods

There are multiple CNN architectures that have been developed for image recognition. In this paper, We adopt models, including LeNet, ShuffleNet, MobileNet, VGG, and Resnet, as our baseline models. In addition, the Transformer was also adopted as an alternative to conventional CNN architectures. All the baseline models are used from MMClassification, an open-source project that is contributed by

3.1 ShuffleNet-v1

ShuffleNet is an advanced-level model of group convolution. By adding a shuffle channel, the information between batches is enabled to flow, which generates the global features of the images for the model.[1] ShuffleNet is an advanced-level model of group convolution. By adding a shuffle channel, the information between batches is enabled to flow, which, Therefore, ShuffleNet can also yield a high performance.

3.2 MobileNet-v2

It is an effective network architecture with a set of two hyper-parameters for the purpose of building tiny, low-latency models that can be easily tailored to the design requirements for mobile and embedded vision applications.[2]

3.3 VGG-16

VGG has six kinds of different structures and it always stands for VGG-16, which has 13 Convolution Layers and 3 Fully Connected Layers. Due to its regular design and simple stackable convolution blocks, it performs well on many datasets [3].

3.4 ResNet-50

Residual Networks(ResNet) learn residual functions based on the inputs from the layers. They build networks by piling residual blocks on top of one another.[4] From the result of empirical research, these networks are simpler to optimize while they can improve accuracy despite the increase in depth.[4] One of the biggest advantages of Resnet is that it successfully handles the issue of vanishing gradient by ‘skipping connection’.[5]

3.5 Transformer-base

The transformer, which is different from the previous models, adopts the mechanism of self-attention and detects the interaction between tokens separated from the images. This architecture contains an encoder, attention blocks, and the head using softmax for classification. The attention mechanism, which realizes the global information flow, leads to a remarkable performance in the aspect of image classification. Many pre-trained models such as GPT-3 BERT, and XLNet, demonstrate the

competitiveness of the transformer compared to conventional CNN architecture.

4. Baseline Models

4.1 Model Choice

As there are various architectures provided which contain many kinds of batch sizes and two basic datasets (ImageNet and CIFAR), 3 models are randomly selected from each architecture. Apart from these two random variables that were used in training, the team controlled epoch number and learning rate to decrease the bias in the comparison.

The result of the model with the highest Top1 accuracy in these five architectures is presented in Figure.1, after setting the epoch number to 300 and the learning rate to 10^{-2} . Although it has large randomness in selecting models from architectures, it is observed that ResNet has much greater accuracy with the same number of epochs than other architectures. Hence, ResNet was decided to be the baseline model for further improvement. The detailed structure of these architectures can be found in Appendix.1.

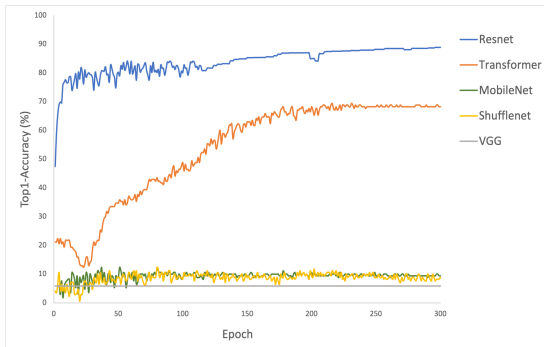


Figure.1 Comparison of Architecture

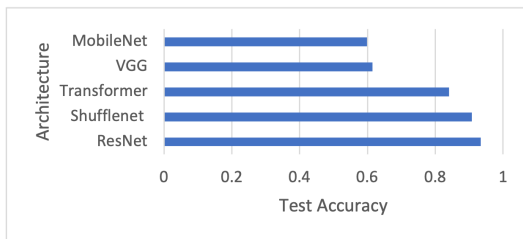


Figure.2 Test Accuracy of Different Architecture

Because there are multiple versions of ResNet, any further comparison had to be done between the different versions that were available, such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152 with setting the batch size to 32, the training dataset to be Imagenet, the epoch number to be 100, and the learning rate to be 10^{-2} . Results are presented below in Figure.3.

As indicated by the figure, with the controlled variables strictly fixed, ResNet-50 appears to have the highest accuracy compared to other versions of ResNet at the same epoch. Therefore, the final baseline model adopted was ResNet-50.

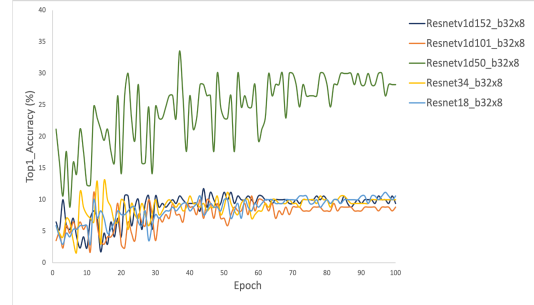


Figure.3 Comparison of Different ResNet Versions

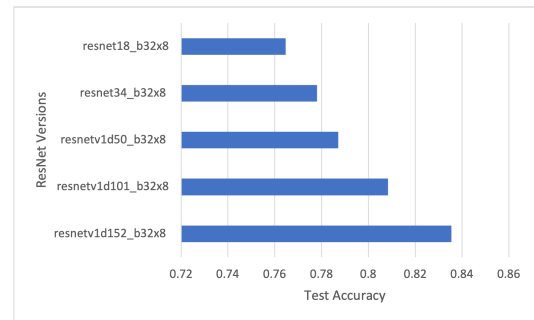


Figure.4 Test accuracy of Different ResNet Versions

4.2 Improvement Technique and Dataset Selection

For the adopted baseline model, ResNet-50, we found a technique called Mixup for data augmentation that can increase the accuracy.[6] In this study, the accuracy increases by 1.2~1.5% on ImageNet-2012 and 1.0~1.4% on CIFAR-10. As the improvement difference between these two datasets is really small and ImageNet significantly uses less time on training than CIFAR through the experience we got in 4.1, so we decided to test this method on ImageNet at the beginning and generalize to CIFAR-10 & CIFAR-100 if it performs well.

The comparison was done between different approaches with training accuracy against epoch. Just as Figure.5 and Table.1 show, there is a non-significant increase after adopting Mixup techniques. So for convenience, the team decided to train the model directly.

	Resnet50 imagenet	Resnet50 mixup_imagenet
Accuracy	0.09433	0.09515
Improvement	-	0.8725%

Table.1 Accuracy and Improvement comparison (epoch = 100)

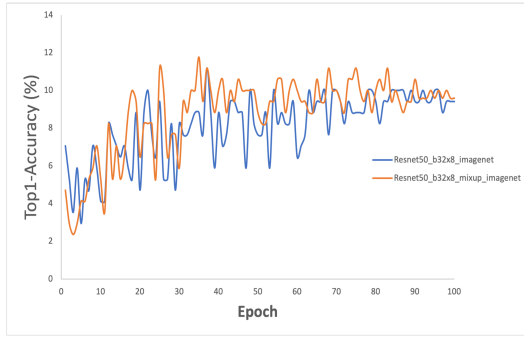


Figure.5 Comparison of Using Mixup or not

Figure 6. shows that there is a considerable improvement when we use the pre-train ResNet which has been trained by the Cifar-10 dataset. It is proven that the model has benefited from transfer learning.

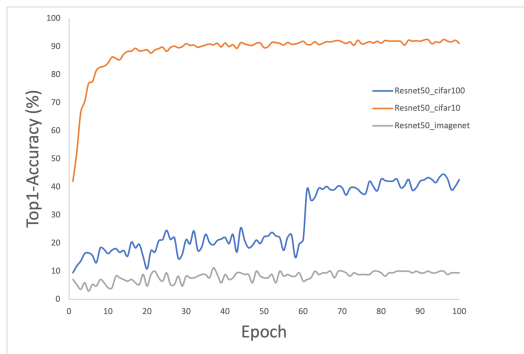


Figure.6 Comparison of Different Datasets

4.3 Learning Rate Tuning

In order to make the model we choose to have a better performance on both the training and testing datasets, we classify the learning rate into two kinds.

1. Non-linear learning rate: Warmup[7] and Cosine Anneal.
2. Constant learning rate: 10^{-1} , 10^{-2} , 2×10^{-2} , and 10^{-3} .

Results are shown in Figure.7.

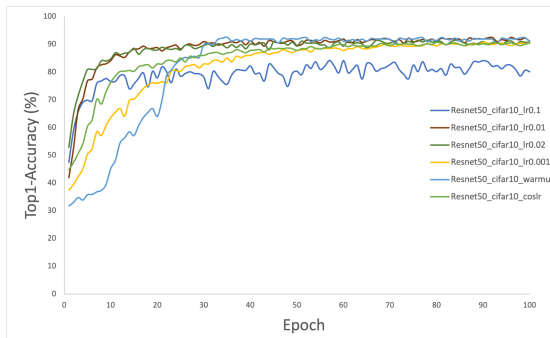


Figure.7 Comparison of different learning rates

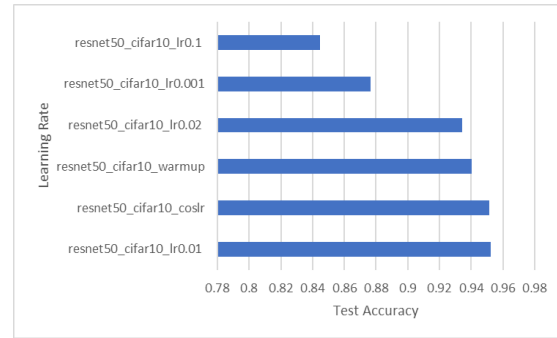


Figure.8 Test Accuracy of different learning rates

The accuracy of testing data when the learning rate is equal to 10^{-2} (is nearly same as using warmup).

4.4 Optimizer Choice and weight decay tuning

The most popular and common way to perform optimization is gradient descent. To improve the accuracy of our model, we decided to try four gradient descent optimization algorithms: SGD, Adagrad, Adadelta, and Adam.

By performing frequent updates with high variance, stochastic gradient descent (SGD)[9] makes the model much faster and can be used online for learning. In Adagrad[9], parameters that perform well in sparse data are selected based on their learning rate. Unlike Adagrad, Adadelta[9] sets restrictions on the window of accumulated gradients rather than grabbing them all. Using an exponentially decaying gradient average, Adam[9] calculates learning rates for parameters.

We have tested these four optimizers based on Resnet50_cifar10_lr0.01, with results shown in Figure.9.

To further improve the accuracy of our model, we have explored the effect of learning rate on four gradient descent optimization algorithms.

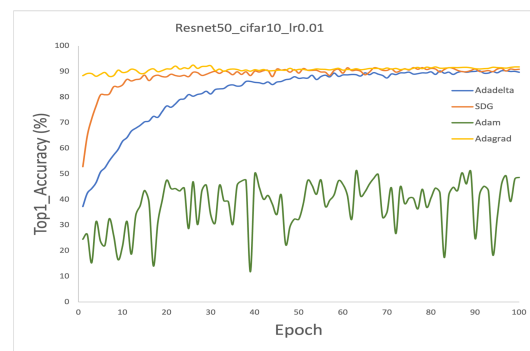


Figure.9 Test Accuracy of different optimizer choice

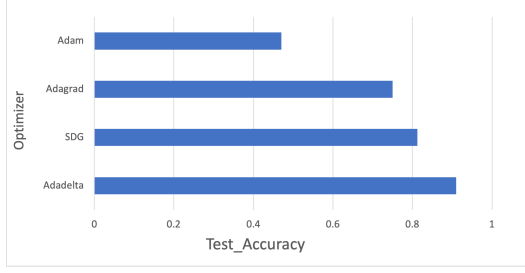


Figure.10 Train Accuracy of different optimizer choice

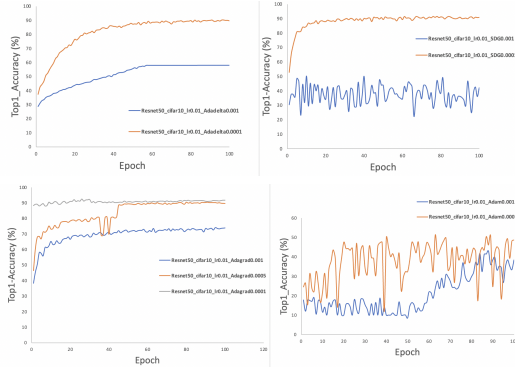


Figure.11 Test Accuracy of different optimizer choice with different learning rate

According to the results show in Figure.11, we find the best model is:

Resnet50_cifar10_lr0.01_Adagrad

5. Result and Discussion

5.1 Best Model

Through Section 4, we can conclude that the best model is ResNet50 trained by CIFAR-10, learning rate equal to 10^{-2} , choosing Adadelta as an optimizer, and weight decay is 10^{-3} .

5.2 Mixup

Mixup was brought forward by Zhang and it is a data augmentation technique that generates a weighted combination of random image pairs from the training data. Given two images and their ground truth labels: (x_i, y_i) , (x, y) , a synthetic training example (\hat{x}, \hat{y}) is generated as

$$\hat{x} = \lambda x_i + (1 - \lambda)x$$

$$\hat{y} = \lambda y_i + (1 - \lambda)y$$

where $\lambda \sim \text{Beta}(\alpha = 0.5)$ is independently sampled for each augmented example[6]. Our result is similar to it.

5.3 Warmup and Cosine Anneal

When using SGD as the optimizer to train neural networks, using a higher learning rate at the initial stage and switching to a lower learning rate later is effective in practice. However, warmup contradicts this approach. It initiates with a small study rate and quickly changes into a large one. Nitish Shirish Keskar's work (Figure.12) shows that learning rate warmup primarily limits weight changes in the deeper layers and that freezing them achieves similar outcomes as warmup[7]. So we guess that warmup is helpful in slowing down the over-fitting phenomenon of mini-batch in the initial stage and keeping the distribution stable. Besides, it may also help to maintain the stability of the deep layer of the model.

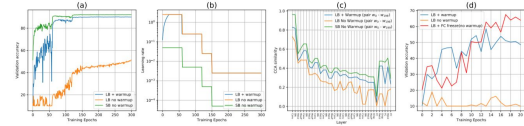


Figure 4: (a) Validation accuracy and (b) Learning rate for the three training setups (c) CCA similarity for i -th layer from two different iterations (0-th (before warmup) and 200-th (after warmup) during training (d) Comparing warmup and FC freezing strategies on VGG11 training

Figure.12 Nitish Shirish Keskar's work

Cosine Anneal is a common approach for adjusting the learning rate. Here is the formula.

$$\eta = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\frac{T}{T} \pi))$$

Hint: η_{\max} is the maximum of learning rate

η_{\min} is the minimum of learning rate

T is the current round

T is the number of half cycle

5.4 CIFAR-10

Compared with CIFAR-100, the number of training images for one category in CIFAR-10 is 10 times bigger which can lead to better training accuracy. Although the CIFAR-10 does not contain any image of flowers, the distribution of the pixels is similar, which can allow the model to repurpose to recognize the flower. This optimization technique makes it possible to model the second task quickly while yielding a better performance[8].

5.5 Adadelta

Adadelta dynamically adapts over time, unlike SGD, utilizing only first-order information, with far less computational cost than simple stochastic gradient descent.[9] The method appears to be robust to noisy gradient information, varied model architectural choices, diverse data modalities, and selection of hyperparameters and requires no manual adjusting of a learning rate.

6. Evaluation

6.1 Difficulty encountered

At the initial stage of our project, we were surprised to find that some models had extremely high accuracy (with top 1 accuracy above 80%). In contrast, some models had only 20% approximately. Moreover, some of those models cost much more time on dataset training, especially for VGG models. It usually took 12 minutes for only one epoch when running the “train.py” file. Therefore, ensuring network stability for 5-6 hours was brutal, which is a requirement difficult to achieve. After trial and error, we made some deletions on unnecessary tensor boards and some adjustments on checkpoint saving intervals, which helped our team to save GPU memory and do better output visualization.

Due to the limited resources and project scope, there still exists some unresolved problems. Before we trained different models, our general understanding of each model was in a very chaotic state. As a result, there was great randomness in model selection when we compared the model due to the limited provided models. For example, when the given architecture (ResNet50) was fixed, we were surprised to find that there exists a significant difference between different datasets (CIFAR and ImageNet). Furthermore, after visualizing each output in one graph, we found that we couldn't converge batch size for different datasets since only “b16x8” was provided for CIFAR-10 training. Therefore, we were forced to compare each batch size on ImageNet.

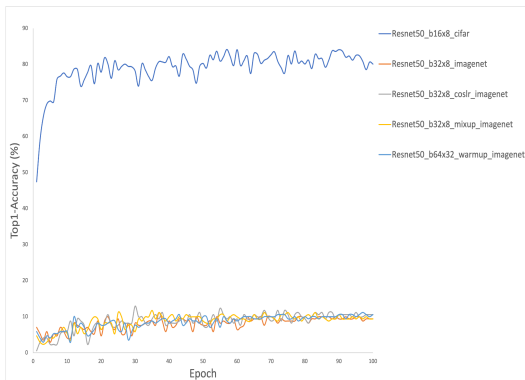


Figure.14 Comparison of different batch size

6.2 Limitation in related work

For experimental purposes, only flower data are included in the training process. It means that the findings may be applicable to flower classification while performance is not satisfying when dealing with data of other creatures or objects. There should be further investigation on other datasets to validate the findings

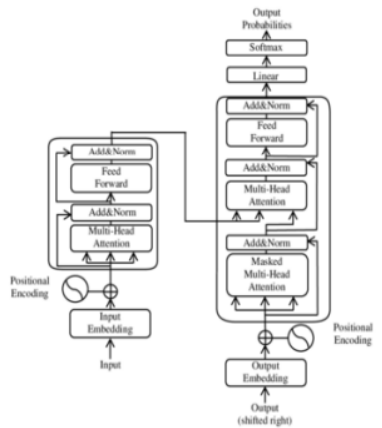
It is aimed to apply our findings in a real scenario. For example, They can be applied to Medical image classification. However, while deep neural networks have shown extraordinary success (often beyond the capability of humans) in tackling complex problems, recent studies have revealed that they are susceptible to adversarial attacks in the form of minor input perturbations that cause a model to predict wrong outcomes.[11] In this paper, we fail to use defense techniques to improve the robustness of the model. It is proposed to try model augmentation. We add an additional class to the possibly targeted neural network models, where the models are trained to categorize all the adversarial samples.

Another limitation is the limited models we used. This paper is designed to compare various models and get an optimal model with the best setting. While the computation resources and time are limited, it is difficult to exhaust all the possibilities. There are more advanced architectures, such as Transformer-XL, and ResNeXt, that are not in the investigated scope. It is hoped that there can be follow-up research to supplement the information missing here. Also, in the process, our team investigates each hyperparameter to find the best setting. By doing this, we aimed to conduct an analysis of the hyperparameter's effect on the final results. If possible, another approach - examine all the combinations of hyperparameters as what the grid search algorithm does - to get an optimal setting.

References

1. Zhang, X., Zhou, X., Lin, M., & Sun, J. (1970, January 1). *ShuffleNet: An extremely efficient convolutional neural network for mobile devices*.
2. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017, April 17). *MobileNets: Efficient convolutional neural networks for Mobile Vision Applications*.
3. Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 1409.1556, 2014.
4. Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, *Deep Residual Learning for Image Recognition*, *arXiv.org*, 2022. <https://arxiv.org/abs/1512.03385>.
5. Dwivedi, P. (2022). *Understanding and Coding a ResNet in Keras*, Medium, 2022. <https://towardsdatascience.com/understanding-a-nd-coding-a-resnet-in-keras-446d7ff84d33>.
6. Zhang, Hongyi, *et al.* (2017). *Mixup: Beyond Empirical Risk Minimization*. "1710.09412.
7. Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong & Richard Socher. (2018). *A Closer Look at Deep Learning Heuristics: Learning Rate Restarts, Warmup and Discussion*. 1810.13243v1.
8. Olivas, S. E. (2010). Chapter 11: Transfer Learning, Handbook of Research on Machine Learning Applications, 2009. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, methods, and Techniques* (Vol. 2). essay, Information Science Reference.
9. Matthew, D. Zeiler. *et al.* (2012). *Adadelata: An Adaptive Learning Rate Method*, 1212.5701v1.
10. Bello, I. *et al.* (2021). *Revisiting resnets: Improved Training and Scaling Strategies*, <https://arxiv.org/abs/2103.07579>
11. Akhtar, Naveed, and Ajmal Mian. "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey." *IEEE Access*, vol. 6, 2018, pp. 14410–14430., <https://doi.org/10.1109/access.2018.2807385>.

Appendices

	3.1 ShuffleNet-v1		3.2 MobileNet-v2		3.3 VGG-16		3.4 ResNet-50		3.5 Transformer-base	
	Image	224×224×3	Image	224×224×3	Image	224×224×3	Image	224×224×3	Image	224×224×3
	Conv1, 3×3 MaxPool, 3×3	112×112×24 56×56×24	Conv1	112×112×32	2× Conv, 3×3 MaxPool, 3×3	224×224×64 112×112×64	Conv1 MaxPool 3×3, s=2	112×112×3		
	Stage2 Stride=2, Repeat=1	28×28×144	Bottleneck1 Bottleneck2 Bottleneck3	112×112×16 56×56×24 28×28×32	2× Conv, 3×3 MaxPool, 3×3	112×112×128 56×56×128	Conv2 × (1×1, 64 1×1, 64 1×1, 256) ×3	56×56×256		
	Stride=1, Repeat=3	28×28×144	Bottleneck4 Bottleneck5 Bottleneck6 Bottleneck7	14×14×64 14×14×96 7×7×160 7×7×320	2× Conv, 3×3 MaxPool, 3×3	56×56×256 28×28×256	Conv3 × (1×1, 128 3×3, 128 1×1, 512) ×4	28×28×512		
	Stage3 Stride=2, Repeat=1	14×14×288	Conv2, 1×1	7×7×1280	3× Conv, 3×3 MaxPool, 3×3	28×28×512 14×14×512	Conv4 × (1×1, 256 3×3, 256 1×1, 512) ×6	14×14×1024		
	Stride=1, Repeat=7	14×14×288			3× Conv, 3×3 MaxPool, 3×3	14×14×512 7×7×512	Conv5 × (1×1, 512 3×3, 512 1×1, 2048) ×3	7×7×2048		
	Stage2 Stride=2, Repeat=1	7×7×576	AvgPool 7×7	1×1×1280	FC1	25088	AvgPool	1		
	Stride=1, Repeat=3	7×7×576			FC2	4096	+ FC			
	GlobalPool 7×7	1×1	Conv3 1×1	1×1×k	FC3	4096	+ Softmax			
	FC	1	FC	1	FC + Softmax	1				
FLOPs:	146,000,000		319,000,000		19,670,000,000		2,520,000,000		1,809,900,000	
Parameters:	1,870,000		3,500,000		143,670,000		42,510,000		86,000,000	

Appendix.1 Baseline Model Analysis