# Convolutional Neural Network Batch pruning via distinctiveness, for fine-grained image classification on Vehicle-X

John Yoo

Research School Of Computing, Australian National University, Canberra, Australia
John.Yoo@anu.edu.au

**Abstract.** Fine-grained image classification is an important task with applications in fields such as object detection and species identification. Networks trained for fine-grained image classification are large and consume exorbitant amounts of computational resources due to the nature of the task. In this paper we expand upon the usage of a convolutional neural network for a fine-grained image classification task using the synthetic dataset Vehicle-X. We apply network reduction and pruning methods to the fully connected (FC) layer to explore the potential for such networks to be limited in their computational expense. In this paper we propose a batch pruning method to achieve more robust pruning when compared to previous research. We find a 'balanced' result with batch pruning that can reduce the neurons in the fully connected layer of a ResNet model by 50% for a reduction in the computational complexity by 1.18% with a sacrifice of 21.37% classification accuracy and thus find the pruning of FC layers for a CNN in our task and approach to be inefficient.

**Keywords:** Convolutional Neural Networks · Pruning · Fine-grained Classification · Network Reduction.

## 1 Background and Introduction

Artificial neural networks and deep learning have proven to be able to produce solutions to many problems in various domains due to their ability to learn high-level abstractions in data through the use of hierarchical architectures. When training such networks, often times they are over-parametrised with large numbers of redundant neurons far larger than the optimal amount. The optimal number of neurons in a neural network are difficult to determine and thus oversized networks are often used. Oversized networks are undesirable as they consume more space and require more computational resources to run. A common approach to this issue is to use methods such as Dropout [2] during training to reduce the number of neurons that influence the model's predictive ability thus aiding in preventing overfitting. Such regularisation methods, however, do not reduce the computation time during evaluation and are thus only useful during the training process.

In this paper we develop a fine-grained classification task to classify samples from the synthetic dataset Vehicle-X [1] to 11 different overarching vehicle types. We use a modified ResNet-18[8] model pretrained on ImageNet as our classifier. Modifying the feed forward fully connected layer of the ResNet model we explore this fine-grained image classification task to expand upon the ideology of utilising the weight matrix for pruning in [4]. [4] Concludes that the static weight matrix is insufficient to determine the functionality of hidden neurons for pruning, and so we conduct experiments while fine-tuning the network after each pruning iteration. In particular, we explore the effects of the distinctiveness pruning method as described in [4] and [5] on the task for the fully connected layer.

### 1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) were first introduced over 30 years ago with LeNet which could recognise handwritten digits [9]. Inspired by biological processes of the animal visual cortex, CNNs are regularised multilayer perceptrons that take advantage of hierarchical patterns in data to assemble more complex patterns using smaller and simpler patterns. Only with recent improvements in computer hardware and development of learning algorithms for network construction have truly "deep" CNNs come into light. CNNs have been successful in a variety of pattern and image recognition applications such as, gesture recognition [10], face recognition [11] and object classification [12]. A key factor to the success of CNNs are the large open source labelled datasets consisting of millions of images available for experimentation and research such as ImageNet, CIFAR-10, CIFAR-100 and MNIST. CNNs contain convolutional layers, pooling layers and fully connected layers. Convolutional layers apply a convolution operation to the input and bring all the information in its receptive field into a single pixel. Pooling layers reduce the spatial size of the feature maps, through this manner it summarises the features present in a region of the feature map generated by a convolutional layer. A feed forward fully connected layer is used to process the final high level features produced by the pooling and convolution layers.

### 1.2 Fine-grained Classification

Fine-grained image classification focuses on differentiating between objects that have small differences and potentially a large number of similarities, e.g., species of insects, animals and car models. A key factor in fine-grained classification is for the model to learn this relevant small difference in order to successfully predict the correct classes. In order to achieve this, large CNNs with many layers are often used to extract the fine details in the input images and thus, for applications where successful models need to be run many times the size of the models leads to large computational resource expenses. In many popular CNN architectures often included as the final layer is a fully-connected layer consisting of many hundreds or thousands of neurons. For E.g. the original ResNet-18 has 16 convolution layers along with 1 MaxPool and 1 Average Pool Layer with a final fully-connected layer consisting of 1000 hidden neurons.

## 2 Methodology

### 2.1 Data and pre-processing

Vehicle-X is a large-scale synthetic dataset that contains 1362 different vehicles of 3D models with an expansive list of editable attributes. The nature of the dataset allows for an unbounded number of samples to be generated. In the original paper [1] a re-id task was explored with the use of Vehicle-X synthetic data combined with real world data. In our task we will only consider a subset of the synthetic data and explore classification on the higher level of 11 different types of vehicles. A total of 75,516 images generated using the Vehicle-X engine are fed through the ResNet model that has been pre-trained on ImageNet. We split the data into Train, Validation and Test sets as described in Table 1.

**Table 1.** Split of dataset into train, validation and testing sets

| **Split** | **# of samples** | **% of dataset** |
|---|---|---|
| Train | 45438 | 60.17% |
| Validation | 15142 | 20.05% |
| Test | 14936 | 19.78% |

Each image in the dataset is 256 x 256 pixels and consists of 3 channels (RGB) an example is given in Figure 1. As our task is a fine-grained image classification task and we are only using a subset of the original dataset we apply various data augmentation methods to attempt to improve the generalisation of our ResNet model. We apply Random Resize Crop which crops the image at a random location and resizes it to the required input size, we also apply Random Horizontal Flip which with a 50% flips the image horizontally. The data augmentations are applied in-place so the number of actual training images is not increased, however, with the augmentations we aim to provide the model with training images that provide a variety of differing viewpoints of the vehicles in order to improve its generalisation ability.

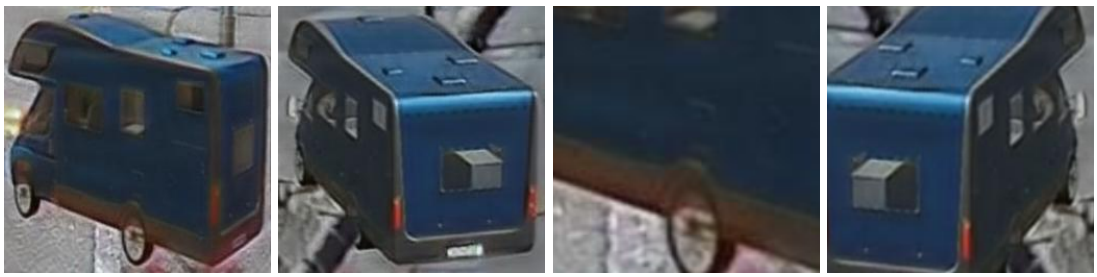

**Fig. 1.** Samples of Vehicle-X generated images [7], note the differing vehicle types, camera angles and lighting. The first 2 images on the left are original training images, the two on the right are results after data augmentation, the first is an example of Random Resize Crop and the second; Random Horizontal Flip

Following this, we convert the image to a tensor and normalise to transform the images such that the mean and standard deviation of the image become 0 and 1 respectively. Normalising the images reduces the skewness and constricts the data to a standardised range which helps the model learn faster and better.

Table 2 illustrates the split of class data in train, validation and test datasets. We can see that all classes are split with similar portions in each of the sub datasets.

**Table 2.** Split of the 11 vehicle types across the 3 sub datasets.

| **Class** | **% in Train** | **% in Validation** | **% in Test** |
|---|---|---|---|
| Sedan | 21.15% | 20.875% | 21.83% |
| SUV | 10.95% | 10.63% | 11.39% |
| Van | 4.34% | 4.29% | 4.22% |
| Hatchback | 20% | 19.32% | 20.38% |
| Mpv | 1.16% | 0.97% | 1.07% |
| Pickup | 3.92% | 3.58% | 3.92% |
| Bus | 5.60% | 5.42% | 5.15% |
| Truck | 7.55% | 7.87% | 7.97% |
| Estate | 4% | 3.99% | 4.13% |
| Sportscar | 16.64% | 17.03% | 16.48% |
| RV | 4.60% | 4.64% | 4.82% |

### 2.2 Evaluation Metrics

To evaluate the performance of our model we first use the classification accuracy on the validation set to tune the hyper-parameters of our baseline model and record the classification accuracy on the test set. To measure the performance of the pruning method and the pruned network we use the classification accuracy on the test set of 15,142 images as well as the number of FLOPs (Floating Point Operations) from one evaluation of the network. FLOPs give us an indicator of the pruned networks computational performance that is not biased by any specific computer hardware.

### 2.3 Baseline Model

We first develop a base CNN to solve the classification task. We use a modified version of ResNet-18 with 500 hidden neurons and 1 hidden layer in the fully connected layers. From a selection of parameters we tune the model on the validation set of 14,936 images. Table 3 details the set of hyper-parameters that produce the highest classification accuracy for the validation set and these are used for the base model.

**Table 3.** Representation of the network, the first 8 are hyper-parameters used to train our base model

| **Attribute** | **Value** | **Description** |
|---|---|---|
| Learning Rate | 0.001 | Networks rate of learning |
| Hidden Neurons | 500 Neurons and 1 layer | Number of layers and number of neurons in each layer |
| Dropout | 0.5 in each layer | Percentage dropout of neurons after each layer |
| Epochs | 50 | Number of training epochs |
| Bias | True | Learn the bias in the network |
| Activation Function | ReLu | Activation function to use in the network |
| Batch Size | 64 | Number of training examples used in one iteration |
| Optimiser | SGD | Algorithm to change weights and reduce loss |
| Input Size | 224 x 224 | Input size of each image |
| Output Size | 11 | Output size equal to the number of output classes |

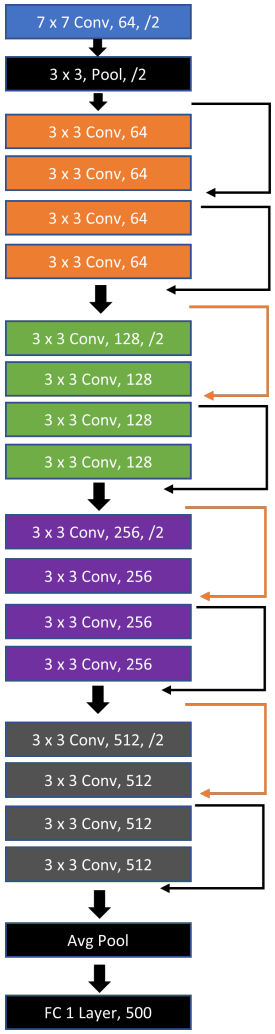


**Fig. 2.** Illustration of the modified ResNet-18 model taking as input a 224 x 224 image, orange lines indicate skip-connections.

### 2.4 Fine-tuning

Fine tuning is a common method used in transfer learning to apply the 'knowledge' one model has attained to another model. In doing so, fine tuning can re-train the weights in the pre-trained model allowing for a quicker convergence and potential increase in accuracy. In our research we fine tune our model to re-train the weights after pruning as a way to allow the model to make up for the lost neurons and to expand upon the findings of [4] which indicate that the static weight matrix is insufficient.

### 2.5 Pruning

Pruning is a concept to reduce the size of a neural network through compression. The importance of each neuron is determined through some ranking method and the least important neurons are removed. This method of network compression was first explored by LeCun et al. in [6] where network compression resulted in better generalisation, fewer training examples required and improved speed of classification.

Distinctiveness pruning is a method whereby the least distinct redundant neurons are pruned. The conceptual idea is that we can calculate the angle between two hidden neurons and determine their degree of similarity. In the original paper [4] and in [5] neurons with similarity of 15 degrees or lower are considered similar, one is added to the other and removed, at the same time where the angle is over 165 degrees the 2 neurons are regarded as complementary and so both are removed. As the model loses redundant neurons the model is compressed and this results in a smaller model and improved computational speed for post-hoc evaluation. At the same time, as the

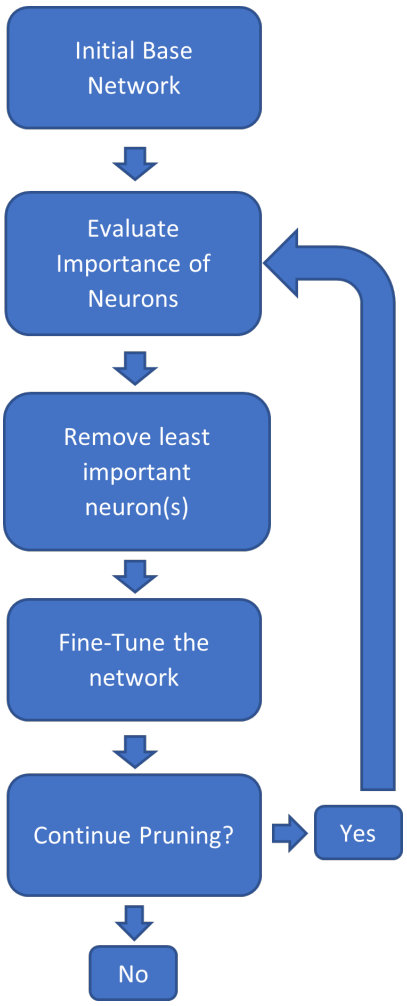


**Fig. 3.** Flowchart of iterative pruning process

model is losing neurons it may also lose predictive ability as weights are discarded. Thus, reaching a compromise between the two aspects is crucial in exploiting the benefits of pruning with manageable accuracy loss.

To compute the distinctiveness of neurons we first normalise the hidden neurons to fit the range of 0-180 (1) and calculate the angle between each neuron pair as in equation (2).

$$\mathbf{x} = \mathbf{x} - mean(\mathbf{x}) \tag{1}$$

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{x}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{y}_i)^2}} \tag{2}$$

For our research purposes as we are exploring the task of fine-grained image classification the feature space is magnitudes larger than of those tasks explored previously in [4,5]. As a result, the number of hidden neurons required to train a base model to learn the representations in the data is also high. Upon initial experimentation of the base model, we find that there are no neurons with angles less than 15 or more than 165. As there were no complementary neurons detected in our model for our task we consider an approach for similarity. To remove X number of similar neurons by determining most distinctively similar hidden neuron pairs, adding one neuron to the other and removing it. In our experiments we choose X to be 50 and fine-tune the model for 25 epochs afterwards, part of the reason for the choice of these 2 parameters is the computational expense as fine-tuning with each prune iteration is expensive for a CNN. Following the basic outline of the flowchart in Fig.3 we:

1. Train the initial baseline modified ResNet-18 model with tuned hyper-parameters until convergence.
2. Apply pruning operations to the fully connected layer.
3. Fine-tune the network with the updated FC weight matrix.
4. Repeat steps 2 and 3 until we are done pruning.

## 3 Results and Discussion

We obtain a final test accuracy of 96.45% using our baseline model. Comparing this to the 1362 class classification problem in the original Vehicle-X paper which achieves 94.34% we can see a comparable high accuracy with our 11 higher level class task. Thus, we can determine that our model and approach is suitable for the 11-vehicle type classification task. Comparing this to our previous results in [13] with a basic neural network model using feature extracted elements of the vehicles which achieved an accuracy of 46.988% we can see that directly using the CNN architecture while computationally expensive, leads to a significant increase in classification accuracy on the same dataset.

**Table 4.** Accuracy and FLOPs of our previous basic neural network approach and our CNN model

| **Model** | **Accuracy(%)** | **FLOPs** |
|---|---|---|
| Baseline Modified ResNet-18 | 96.45 | 13,782,396 |
| Basic Neural Net | 46.99 | 1,030,500 |

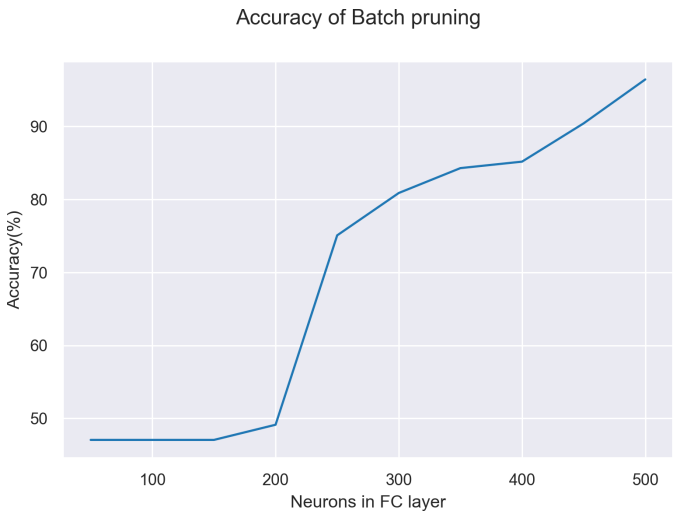


**Fig. 4.** Plot of accuracy vs number of neurons remaining in the hidden layer of the FC

We apply our pruning approach following the overall methodology in Fig.3 and iteratively prune until there are no more extra neurons to prune from the fully connected layer. The accuracy against the test set is plotted against the number of neurons pruned from the model.

In Fig.4 we observe a common trend of extreme decrease at some point in testing accuracy in our pruning method which is expected as the process would eventually remove important neurons. After this inflection the model has an extremely low classification accuracy that is consistent until all possible neuron batches are removed. An interesting observation can be seen at which point in the pruning process each method experiences the inflection. This point of inflection can be noted as the point at which the model loses the majority of the neurons required to successfully predict. We propose the usage of this as an indicator for the robustness of each pruning method. We will refer to this as the inflection point, to describe the point where the model loses most of its predictive ability and the point where the model cannot be pruned anymore in order to be able to have a decent predictive ability on the dataset. Table 5 lists the number of FLOPs in the CNN with each iteration of 50 neuron pruning. We note that the total number of FLOPs that can be reduced in this manner is 292,950 or 2.125% of the entire CNN model. Table 6 illustrates methods of importance. In particular, we compare the baseline ResNet-18 model with the pruned ResNet on the inflection point, there is a drop of 21.37% in accuracy, a reduction of half the hidden neurons in the FC layer and subsequently a reduction of 162,750 FLOPs which corresponds to a 1.18% reduction in the models computational expense. This large drop in accuracy can be expected as half the hidden neurons in the FC layer were pruned from an already tuned FC layer. We find that the reduction in computational expense versus the reduction in classification accuracy is imbalanced and does not justify the usage of distinctiveness pruning on the FC layer of ResNet.

**Table 5.** Number of neurons in the FC layer and the corresponding number of FLOPs in the entire model

| **Neurons in FC Layer** | **FLOPs** |
|---|---|
| 500 | 13,782,396 |
| 450 | 13,749,846 |
| 400 | 13,717,296 |
| 350 | 13,684,746 |
| 300 | 13,652,196 |
| 250 | 13,619,646 |
| 200 | 13,587,096 |
| 150 | 13,554,546 |
| 100 | 13,521,996 |
| 50 | 13,489,446 |

**Table 6.** Models of importance with their accuracies, FLOPs and neurons in the FC Layer

| **Model** | **Accuracy%** | **FLOPs** | **Neurons** |
|---|---|---|---|
| Baseline ResNet-18 | 96.45 | 13,782,396 | 500 |
| Basic Neural Net | 46.99 | 1,030,500 | 500 |
| ResNet-18 + pruning (inflection) | 75.08 | 13,619,646 | 250 |
| ResNet-18 + pruning (best acc) | 90.43 | 13,749,846 | 450 |
| ResNet-18 + pruning (lowest FLOPs) | 47.04 | 13,489,446 | 50 |

## 4 Conclusion and Future Work

In this paper we have explored a fine-grained image classification problem using images generated from the synthetic dataset generator Vehicle-X. We used a modified ResNet-18 CNN model pretrained on ImageNet. We proposed an 11 class vehicle type classification problem with a subset of the original Vehicle-X dataset. We implement an iterative batch pruning method to explore the effects of pruning on the fully connected layer of the CNN. In particular, we explore the distinctiveness pruning method [4, 5] and apply the removal of a batch of 50 similar neurons in each of the 9 iterations. Observations reveal an inflection point that we propose can be used to indicate the robustness of a pruning method. We find that batch pruning on the CNN removes most neurons before the inflection point as compared to traditional methods of similarity and complementary angle removal in distinctiveness

pruning. Our best balanced pruning results indicate with a sacrifice of 21.37% in accuracy we can reduce the evaluation computational expense of the CNN by 1.18%. This imbalanced tradeoff of accuracy for performance is inefficient especially considering the computational resources required to find a balance between neurons to prune and classification accuracy. We observe the following limitations in our approach: Due to the computational expenses of retraining a CNN model multiple times and thus the chosen parameters to prune the model; we have been unable to experiment on differing sizes of batch pruning and thus were not able to confirm that at 250 neurons pruned it was an exact inflection point as in [13] where the fast training afforded us the ability to prune small number of neurons at a time. The fully connected layer is also computationally a relatively small part of the entire CNN model, in our experiments we found that the potential relative gains in computational time is not great compared to the accuracy loss and time to extract pruning data from the model. On this note, the time to prune the fully connected layer in a CNN and fine-tune the entire model is very time-consuming and as there are set rules for how many neurons a task needs, it required extensive experimentation and thus the number of epochs we could afford to spend training the model were not confirmed to be enough to reach convergence for the fine-tuning of the pruned model. We consider the following future works: If sufficient time allows, we would like to conduct more thorough experimentation of the different pruning batch sizes especially starting from smaller amounts of neurons. This would give us a better idea of the exact inflection point for our model and task as our current experiments give us a vague idea of after how many neurons pruned this point occurs. A control group for more initial sizes of hidden neurons, in this paper we only consider 1 baseline model with 500 hidden neurons, increasing the number of neurons exorbitantly and observing the effects of different pruning batch sizes could shed light in an optimal pruning starting point for this task that may differ from what we used. We would also want to explore ways to improve the computational speed of distinctiveness calculation via methods such as parallelisation and multi-threading as this is a core computation in the pruning process and optimisation of this could lead to large amounts of time saved. In this paper we have only explored distinctiveness, in the future we would like to extend the batch pruning of CNNs with various other heuristics, starting with badness and sensitivity and compare the approaches in accuracy, computational time and pruning. We would also like to explore these pruning methods with a CNN architecture on different datasets and tasks, starting with the more complex 1362 class classification task on a larger Vehicle-X dataset and moving onto different domains.

## References

1. Yao, Y., Zheng, L., Yang, X., Naphade, M., & Gedeon, T. (2019). Simulating content consistent vehicle datasets with attribute descent. arXiv preprint arXiv:1912.08855.
2. G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. http://arxiv.org/abs/1207.0580, 2012.
3. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015a.
4. Gedeon, T.D.: Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. In: Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems. pp. 26-29. IEEE (1995).
5. Gedeon, T., Harris, D.: Network reduction techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications. vol. 1, pp. 119-126 (1991)
6. LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In Advances in neural information processing systems (pp. 598-605).
7. Yao, Y., Zheng, L., Yang, X., Naphade, M., & Gedeon, T. (2019). https://github.com/yorkeyao/VehicleX
8. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
9. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989.
10. Bobić, V., Tadić, P., Kvaščev, G. (2016, November) "Hand gesture recognition using neural network based techniques." in Neural Networks and Applications (NEUREL), 2016 13th Symposium on (pp. 1-4). IEEE
11. Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997) "Face recognition: A convolutional neural-network approach."IEEE transactions on neural networks, 8(1):98-113.
12. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ...&Rabinovich, A. (2015)"Going deeper with convolutions." in Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
13. Yoo, John (2021) "Batch pruning, a robust distinctiveness pruning method for fine-grained image classification". 4th ANU Bio-inspired Computing Conference