EFFICIENTNETV2 NEURAL ARCHITECTURE SEARCH

Final Report, November 2022

THE UNIVERSITY OF SYDNEY

Data Science Capstone Project

DATA5709

1. Adam Peaston (308 184 653)

This project was completed with the support of Strong Compute supervisors, and the use of Strong Compute computational resources, for which the author expresses many sincere thanks.

ABSTRACT

Optimising large machine learning models is computationally expensive. A single large neural network with millions or billions of parameters can require weeks of training on specialised hardware. Conducting an effective hyper-parameter space search of large models is simply not feasible with contemporary computational infrastructure. Recent work by [Tan and Le, 2020] and [Tan and Le, 2021] proposed a procedure by which small-scale representative models could be designed via Neural Architecture Search (NAS) on a proxy task and then scaled-up to larger sizes, preserving structural self-similarity, to suit available training and inferential resource constraints and achieve reliable large-scale performance. This work seeks to build on that of [Tan and Le, 2020] and [Tan and Le, 2021] by exploring the use of Bayesian Optimisation for learning a surrogate objective function via Sequential Model-Based Optimisation (SMBO). Where [Tan et al, 2020] uses separate mechanisms for NAS and scaling, (SMBO via controller RNN training for NAS, and then compound scaling via free parameter), this work couples the NAS and scaling mechanisms by describing the search space in terms of numeric parameters that can be directly applied to model scaling. In this work, the model construction procedure (the NAS search space) is predominantly parameterised with numeric ranges to maximise the relative advantage of stochastic exploration of numeric ranges as opposed to discrete architectural choices, using Gaussian Processes as the surrogate objective function.

TABLE OF CONTENTS

1. INTRODUCTION

Training large deep learning models is expensive both in energy resources and time. Further, with the increase in capability and reliability of computer vision models the demand for bespoke models fit for specific domains has increased as organisations discover more and more varied applications for this technology. As a result, focus has shifted to the development of compute-efficient model architectures such as MobileNet [Howard et al, 2017] and MobileNetV2 [Sandler et al, 2019] which are small enough to fit on hand-held devices while achieving near State Of The Art (SOTA) accuracy and reliability.

The efficiency of MobileNet [Howard et al., 2017] was made possible by then-novel convolution operations such as depth-wise separable convolutions. Later development of the Squeeze-and-Excitation operation [Hu et al, 2019] as a parameter-efficient mechanism to preserve channel-wise interdependencies was utilised by EfficientNet [Tan et al, 2020] and EfficientNetV2 [Tan et al, 2021], a family of models discovered by the use of Neural Architecture Search (NAS) to find a strong small-scale baseline model, and a principled compound scaling heuristic for scaling up to achieve higher performance within available training and inferential resource constraints. NAS conducted by [Tan et al, 2020] and [Tan et al, 2021] relied on Recurrent Neural Network (RNN) "controller" models to predict model architectures consisting of discrete architecture decisions.

This work explores the use of Bayesian Optimisation to guide NAS, applying Gaussian Process (GP) surrogate models tuned via Sequential Model-Based Optimisation (SMBO) to predict strong model architectures. The proposed model construction procedure (the NAS search space) is determined by continuous parameters to maximise the advantage of the semi-stochastic Gaussian Process Expected Improvement (GPEI) algorithm for updating the GP surrogate model.

The goal of this work is to demonstrate an approach to NAS whereby a baseline model can be used to craft a search space which can be automatically searched by SMBO of a GP surrogate model to discover more optimal architectures for a given application. Such an approach would significantly ameliorate the otherwise daunting task of searching the vast space of potential architectures for models suited to each novel application.

2. RELATED LITERATURE

A review of some of the relevant literature follows, concerning three core areas underpinning this work; modern computer vision architectures, Neural Architecture Search, and Bayesian Optimisation.

2.1 MODERN COMPUTER VISION ARCHITECTURES

Since the development of AlexNet by [Krizhevsky et al, 2012] and its success in the ImageNet LSVRC-2010 contest, Convolutional Neural Networks (CNN) have predominated computer vision tasks. AlexNet was also early to demonstrate the advantage of distributed training via model parallelism where a single model is split across multiple Graphical Processing Units (GPU). Many challenges of training deep neural networks, including CNNs, were addressed by [Ioffe et al, 2015] with the proposal of Batch Normalization to prevent inter-layer covariance shift and mitigate vanishing gradients, and by [Kaiming et al, 2015] with the demonstration of residual connections (ResNet) which prevents additional layers from degrading model accuracy and smoothes the loss-landscape [Li et al, 2018] resulting in more stable training. ResNet also introduced the architectural paradigm of repeated layer blocks with similar or identical internal structure.

In response to growth in demand for computer vision capability in mobile devices with limited memory and computational resources, MobileNet was developed by [Howard et al, 2017]. This model architecture was governed by a global hyper-parameter that trades-off between latency (at smaller model size) and accuracy (at larger model size). MobileNet also applied depth-wise separable convolution (DWSC) operations, illustrated in Figure 1, to achieve greater parameter efficiency. Standard convolutional operations on an image tensor of size (C,H,W) aggregate pointwise multiplication between N filters of size (C, K, K) and same-size patches of the (possibly padded) input image. Each kernel aggregates information from all channels of the input image. In contrast, DWSC comprises two steps; first aggregating pointwise multiplication between C filters of size (1, K, K) each applied to only a single channel of the input image returning a feature map of size (C,H,W), followed by pointwise multiplications between N filters of size (C, 1, 1) and each spatial location in the feature map.

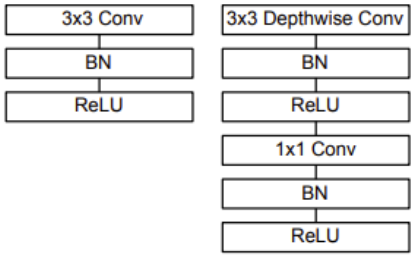


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

Figure 1 - Depth-wise separable convolution operation, reproduced from [Howard et. al. 2017] Figure 3.

This innovation by [Howard et al, 2017] demonstrated that adequate communication of inter-channel information can be achieved without the full (C,K,K) kernel aggregation in a single step. Using DWSC in place of standard convolutions results in $CK^2 + NC$ parameters instead of NCK^2 and correspondingly fewer numeric multiplications when convolving over the input image. With 3×3 kernels, MobileNet reported 8-9 times less computation compared with standard convolutions.

This work was continued by [Sandler et al, 2019] in development of MobileNetV2 which demonstrated several variations on the DWSC operation, reversing the sequence of depthwise and pointwise operations in their "bottleneck with expansion" layer (later referred to as a "Mobile inverted Bottleneck" or "MBConv", refer Figure 2) and finding advantage in omitting non-linearities at the output of this layer.

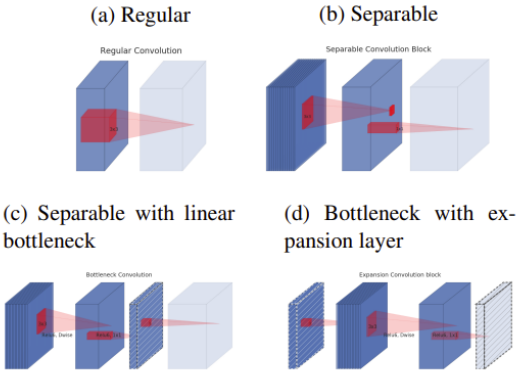


Figure 2: Evolution of separable convolution blocks. The diagonally hatched texture indicates layers that do not contain non-linearities. The last (lightly colored) layer indicates the beginning of the next block. Note: 2d and 2c are equivalent blocks when stacked. Best viewed in color.

Figure 2 - Bottleneck with expansion aka. “MBConv” layer,
reproduced from [Sandler et al, 2019] Figure 2.

Concurrently [Hu et al, 2019] observed that development of convolutional operations had focussed on better utilising spatial correlations, and recognised the opportunity for inter-channel information enhancement with the development of the “Squeeze-and-Excitation” (SE) operation, illustrated in Figure 3. The SE operation average-pools each channel of the (C,H,W) featuremap to produce a (C,1,1) vector which is then passed through a 2-layer feed-forward network, typically a bottleneck shape, and then through a sigmoid nonlinearity. The resulting (C,1,1) vector of sigmoid activations is then multiplied by the (C,H,W) input featuremap to produce a scaled version of the featuremap which can then be passed to the next operation of the layer.

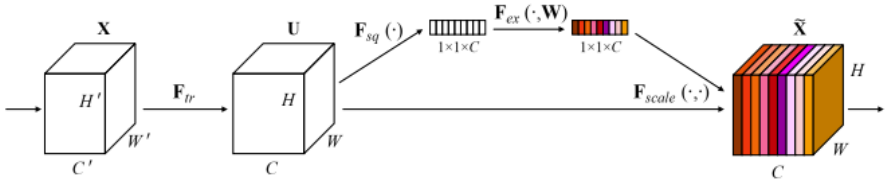


Fig. 1. A Squeeze-and-Excitation block.

Figure 3 - Squeeze-and-Excitation operation,
reproduced from [Hu et al, 2019] Figure 1. A.

The SE operation draws inspiration from the attention gating mechanisms popularised by [Vaswani et al, 2017] and was shown by [Hu et al, 2019] to enhance the performance of ResNet, VGG, and Inception model architectures.

2.2 NEURAL ARCHITECTURE SEARCH

The origin of Neural Architecture Search (NAS) as a concept was necessarily contemporaneous with the development of the neural network family of machine learning models. NAS has since been formalised as a framework for describing procedures to design or discover neural network architectures, categorised by three main considerations as follows.

1. Search space (rules or procedures for generating valid network architectures).
2. Search strategy (the mechanism by which network architectures are sampled from the search space).

3. Objective function (the criteria by which a proposed network architecture is to be evaluated).

A wide variety of approaches and techniques are more or less applicable to each of these considerations depending on task and available resources.

Recent application of NAS to the task of CNN design was demonstrated by [Zoph et al, 2017] with the use of a "controller" Recurrent Neural Network (RNN) to predict sequential model hyper-parameters as a search strategy, as shown in Figure 4. The controller RNN is trained via reinforcement learning by predicting model architectures and receiving reward signals in the form of the objective function results.

This approach is an example of a general procedure within the NAS framework known as Sequential Model-Based Optimisation (SMBO) which refers to the use of a surrogate model which is trained sequentially on samples of the objective function evaluated at points in the search space. The surrogate model learns an estimation of the objective function and is used to sample hyper-parameters from the search space which optimise the objective function. The objective function employed by [Zoph et al, 2017] consisted only of accuracy of the CNN.

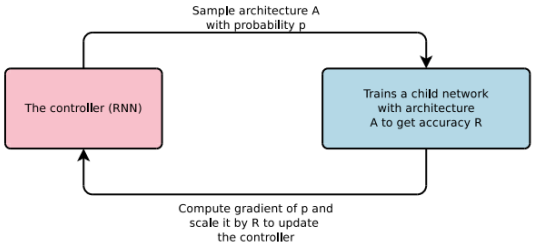


Figure 1: An overview of Neural Architecture Search.

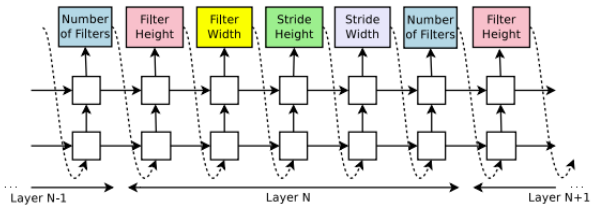


Figure 2: How our controller recurrent neural network samples a simple convolutional network. It predicts filter height, filter width, stride height, stride width, and number of filters for one layer and repeats. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.

Figure 4 - NAS using reinforcement learning search strategy with RNN controller, reproduced from [Zoph et al, 2017] Figure 1 and Figure 2.

The use of reinforcement learning to train an RNN controller model, where each update to the controller requires training a candidate model with hyper-parameters proposed by the controller model, is an approach which requires significant computational resources and time. In order to accelerate the overall NAS procedure, [Zoph et al, 2017] utilised a complex configuration of model parallelism with asynchronous updates.

This work was refined by [Zoph et al, 2018] wherein the task of the controller was simplified to propose the internal structure of two types of convolutional layer or "cell", referred to as "normal cells" and "reduction cells", as shown in Figure 5, which are then stacked to desired model depth. Normal cells return the same shape of featuremap as the input tensor, while the reduction cells return a featuremap tensor in which the spatial resolution is reduced by half.

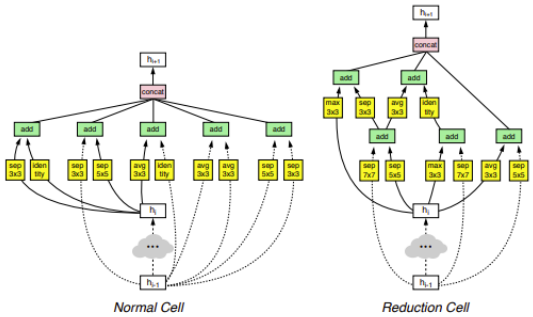


Figure 4. Architecture of the best convolutional cells (NASNet-A) with $B = 5$ blocks identified with CIFAR-10 . The input (white) is the hidden state from previous activations (or input image). The output (pink) is the result of a concatenation operation across all resulting branches. Each convolutional cell is the result of B blocks. A single block is corresponds to two primitive operations (yellow) and a combination operation (green). Note that colors correspond to operations in Figure 3.

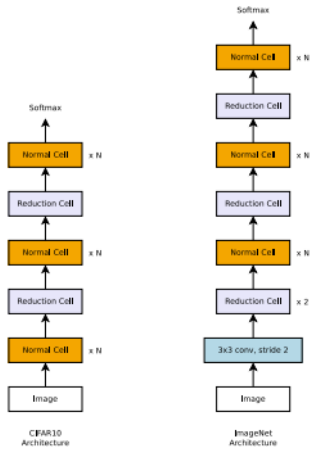


Figure 2. Scalable architectures for image classification consist of two repeated motifs termed *Normal Cell* and *Reduction Cell*. This diagram highlights the model architecture for CIFAR-10 and ImageNet. The choice for the number of times the Normal Cells that gets stacked between reduction cells, N, can vary in our experiments.

Figure 5 - NASNet-A Normal Cells and Reduction Cell, and stacked repeating blocks, reproduced from [Zoph et al, 2018] Figure 4 and Figure 2.

The controller was once again a RNN trained by reinforcement learning. The family of CNNs produced by this means were called NASNet. The overall structure of NASNet is thus characterised by repeated layers of consistent structure. The number of repeats of each layer the authors left as free parameters which the authors tailored to suit the input image size and classification task. The objective function employed by [Zoph et al, 2018] also consisted only of accuracy of the CNN.

A further development of RNN controller model-driven NAS for CNNs was made by [Tan et al, 2019] wherein each block of the CNN (comprising a stack of repeating layers) was searched independently, per Figure 6, by the controller RNN while each constrained to the overall sequence of operations within the MBConv structure by [Sandler et al, 2019].

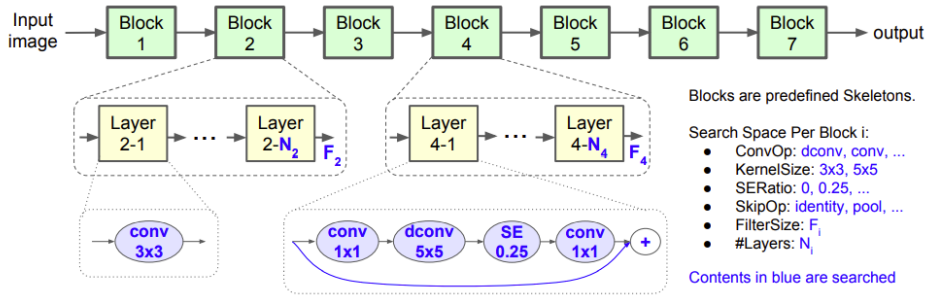


Figure 4: **Factorized Hierarchical Search Space.** Network layers are grouped into a number of predefined skeletons, called blocks, based on their input resolutions and filter sizes. Each block contains a variable number of repeated identical layers where only the first layer has stride 2 if input/output resolutions are different but all other layers have stride 1. For each block, we search for the operations and connections for a single layer and the number of layers N, then the same layer is repeated N times (e.g., Layer 4-1 to 4-N_4 are the same). Layers from different blocks (e.g., Layer 2-1 and 4-1) can be different.

Figure 6 - MnasNet Factored Hierarchical Search Space,
reproduced from [Tan et al, 2019] Figure 4.

The search space comprised model configurations expressed in terms of relative variance from MobileNetV2, for example the number of layers in each block searched from the range {-1, 0, 1} relative to the corresponding block in MobileNetV2. The objective function comprised both accuracy and inference latency on mobile devices. The authors also performed experiments which demonstrated that greater accuracy and reduced latency is achievable with the allowance for diversity between blocks, and that SE operations provide an advantage in terms of enhanced accuracy at comparable latency.

Building on the success of MobileNet and MobileNetV2, and in particular the adaptability of the MBConv layer architecture, the authors of [Tan et al, 2020] focussed on a procedure for small-scale model selection by NAS, coupled with a

principled compound scaling procedure for scaling model size up to suit available hardware resources. The objective function combined accuracy, memory requirement, and Floating Point Operations (FLOPS) for inference as a hardware-agnostic proxy for inference speed. The inclusion of inference FLOPS in the objective function also has the effect of selecting for training speed, though this was not stated explicitly in this work.

The principled scaling procedure, illustrated in Figure 7, called for the selection by NAS, using grid-search, of three hyper-parameters α, β, γ which are used to determine model depth d, with w, and resolution r by means of a free parameter ϕ as follows.

$$d = \alpha^{\phi}, w = \beta^{\phi}, r = \gamma^{\phi}$$
$$s.t. \alpha \times \beta^2 \times \gamma^2 \approx 2$$
$$and\ \alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

The free parameter ϕ thus allows the model designer to determine the approximate increase in FLOPS over the baseline model.

The authors of [Tan et al, 2020] used the same search space as [Tan et al, 2019] which in turn used a MobileNetV2 backbone composed of a stack of blocks, each made up of a number of MBConv layers. The family of CNNs generated by this procedure, referred to as EfficientNet, achieved SOTA performance on the ImageNet image classification task using a fraction of the number of parameters of performance-comparable models.

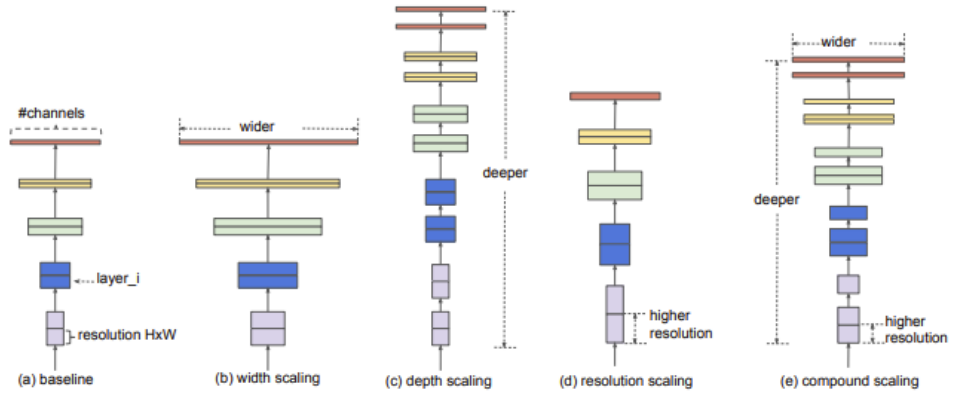


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equation 2.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 7 - Principled compound scaling and EfficientNet-B0 architecture, reproduced from [Tan et al, 2020] Figure 2 and Table 1.

This work was then revisited by [Tan et al, 2021], experimenting with replacement of MBConv layers with a simpler variant referred to as "Fused-MBConv", illustrated in Figure 8, which replaces the DWSC operation with a standard convolution prior to SE and pointwise convolution operations. The authors note that this layer type improves training speed, particularly when implemented in earlier blocks of the model, because MBConv layers "cannot fully utilise modern mobile or server accelerators" [Tan et al, 2021]. If all blocks of the model are replaced with Fused-MBConv layers, however, the resulting increase in FLOPS also increases training and inference time. Therefore, the authors apply NAS to optimize the depth at which Fused-MBConv blocks pass to MBConv blocks.

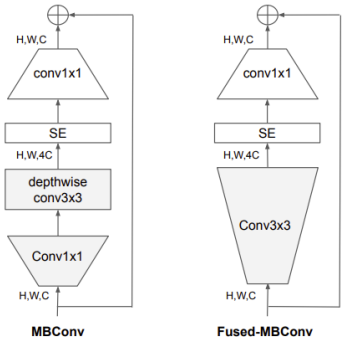


Figure 2. Structure of MBConv and Fused-MBConv.

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

Figure 8 - MBConv and Fused-MBConv layers, and EfficientNetV2-S architecture, reproduced from [Tan et al, 2020] Figure 2 and Table 4.

It should be noted that EfficientNetV2-S (Figure 8) omits the SE component from the Fused-MBConv layer. This is sensible as the ordinary convolutional layer has the capacity to encode inter-layer dependencies.

The NAS implemented by [Tan et al, 2021] considered a search space based on EfficientNet as a backbone, and applied an objective function composed of accuracy, epoch training time, and number of model parameters. The authors of [Tan et al, 2021] applied a similar principled scaling procedure to [Tan et al, 2020]. The maximum inference image size was restricted to a resolution of 480 as larger image sizes resulted in unacceptable memory and training time cost, and more layers were gradually added to later blocks to increase model capacity at minor increase in inference time.

Additional recent work by [Radosavovic et al., 2020] sought to approach NAS by considering the design of the search space and progressively narrowing the search for architectures within that space in a sequential manner. This approach is reminiscent of Bayesian Optimisation insofar as the search space is progressively narrowed to regions of the beginning search space within which higher performance models are likely to be found. Where this work by [Radosavovic et al., 2020] considers a search space that is discrete, and the search space therefore set-like, this project explores techniques for approximating the search space as continuous and therefore efficiently searchable with Bayesian Optimisation guiding SMBO of a GP surrogate model.

2.3 BAYESIAN OPTIMISATION

Reinforcement learning of RNN controller models appears to be the preferred search strategy implemented by the authors in the previous section. These are natural choices for search spaces describable as sequences of tokens representing discrete computational operations and their associated hyper-parameters. However these

approaches present limitations to efficient search of a broader parameter space due to the significant training time necessary to generate data for reinforcement learning of an RNN controller. For a discrete search space, the popular alternative mentioned by [Tan et al, 2021] is random search which the authors of [Zoph et al, 2018] note is a "difficult baseline to beat", particularly with a well-constructed search space.

As pointed out by [Bergstra et al, 2012], random search of continuous spaces has a particular advantage over grid-search. This is because, as shown in Figure 9, the search procedure is able to explore a greater variety of hyper-parameter values for each hyper-parameter searched, and efficiently discover optimal specifications for hyper-parameters which have the greatest impact on the objective function.

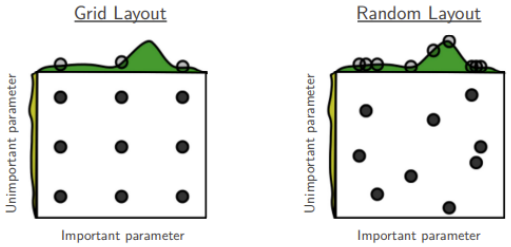


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Figure 9 - Grid-layout versus Random-layout search spaces, reproduced from [Bergstra et al, 2012] Figure 1.

In contrast, grid-search will spend a significant proportion of the time evaluating the objective function for hyper-parameter combinations that are far from the optimal range. An analogous mechanism may be responsible for the strong performance of random search of discrete spaces compared with reinforcement learning noted by [Zoph et al, 2018]. Random search also has the advantage of being parallelizable, where reinforcement learning is necessarily sequential.

An alternative strategy is to implement Sequential Model-Based Optimization (SMBO) using Bayesian Optimisation which seeks to fit a posterior distribution over the objective function (surrogate function, analogous to the controller RNN in reinforcement learning) from evaluations of the objective function. Bayesian Optimisation is advantageous for tasks where the objective function is computationally expensive to evaluate, such as for modern CNNs. Bayesian Optimisation is also able

to take advantage of the relative strength of random search compared with grid-search of continuous spaces by more efficiently sampling from the continuous space, but should be expected to out-perform random search as hyper-parameter proposals improve in quality each iteration.

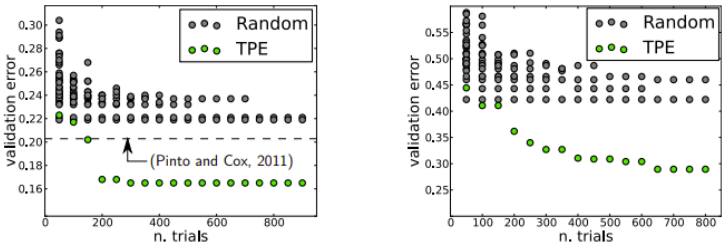
Two model-based methods are explored by [Bergstra et al, 2011] for defining the surrogate function; Gaussian Processes (GP) and Tree-structured Parzen Estimators (TPE). In this work [Bergstra et al, 2011] compares these methods with manual search and random search of hyper-parameters for Deep Belief Networks applied to the "convex" and "MNIST rotated background images" datasets, finding that TPE and GP methods both out-perform manual and random search, with TPE performing best, as shown in Figure 10.

	convex	**MRBI**
TPE	**14.13** $\pm$0.30 %	**44.55** $\pm$0.44%
GP	16.70 $\pm$ 0.32%	47.08 $\pm$ 0.44%
Manual	18.63 $\pm$ 0.34%	47.39 $\pm$ 0.44%
Random	18.97 $\pm$ 0.34 %	50.52 $\pm$ 0.44%

Table 2: The test set classification error of the best model found by each search algorithm on each problem. Each search algorithm was allowed up to 200 trials. The manual searches used 82 trials for **convex** and 27 trials **MRBI**.

Figure 10 - Advantage of TPE algorithm for Deep Belief Networks on two datasets, reproduced from [Bergstra et al, 2011] Table 2.

This work was built on by [Bergstra et al, 2013] applying Bayesian Optimisation to early computer vision model architectures including convolution operations. TPE was compared with random search in selecting settings of 238 hyper-parameters governing the number and type of layers, and layer configuration options, some mutually-exclusive. It was found that TPE significantly outperforms random search in terms of optimisation efficiency and time, and matched the performance of manual-tuning on the CIFAR10 dataset, as shown in Figure 11.



Method (# configurations)	LFW View 2 Error (%)	PubFig83 View 2 Error (%)
TPE-optimized (750)	**15.5** ± .7	**13.50** ± .7
High-throughput (15K)	**15.9** ± .7 (Pinto & Cox, 2011)	14.78 ± .45 (Pinto et al., 2011)
Random search (2K)	20.8 ± .8	19.0 ± .8
Chance	50.0	98.8

Figure 2. Optimization of validation set performance on data sets LFW (top left) and PubFig83 (top right). The TP algorithm finds configurations with significantly better validation set error than a 2000-trial random search or the 15,0 trial random searches carried out in Pinto & Cox (2011) and Pinto et al. (2011). Grey dots in the top panels within column represent the lowest error among T random trials (as T increases to the right); green dots denote the lowest error observed within the first T suggestions by the TPE algorithm. On test data ("view 2", bottom), TPE has discovered the best known model configuration in the search space within 750 trials, but our 2000-trial random search has not come close. View 2 error rates are given with a 95% confidence interval assuming Bernoulli-distributed errors.

Figure 11 - Advantage of TPE algorithm compared with random search for CNN-like architectures, reproduced from [Bergstra et al, 2011] Figure 2.

This work by [Bergstra et al, 2013] also introduced a null or prior distribution specification language which is used to encode prior beliefs about optimal hyper-parameter specifications. The syntax of this language is recognisable in modern software packages such as Ax, and could be used to encode confidence or uncertainty in the hyper-parameter specifications of SOTA architectures found by other means.

3. RESEARCH/PROJECT PROBLEMS

Significant parameter efficiency gains have been demonstrated by [Tan et al, 2020] and [Tan et al, 2021] while also achieving SOTA performance on key computer vision benchmarks such as ImageNet. Parameter efficiency also offers benefits for training time, memory overhead, and inference latency.

The essence of the approach by [Tan et al, 2020], of interest for this work, is the two step methodology of (1) selection of model architectures by NAS with a proxy task as the objective function, then (2) scaling-up the selected model while preserving some degree of self-similarity to suit available resources or deployment context. The scaling heuristic employed by [Tan et al, 2020] and [Tan et al, 2021] is based on achieving predictability of resource requirement. This leads to the first hypothesis for this project as follows.

Hypothesis 1: Alternative approaches to preserving self-similarity in scaling may be advantageous to achieving large scale performance based on proxy task validation.

Principled NAS search strategies such as reinforcement learning or Bayesian Optimisation offer an advantage over grid search and random search in that they are able to more efficiently search the hyper-parameter domain. Furthermore, neural architectures are describable as directed acyclic graphs encoding discrete design decisions such as types of operations and number of layer repeats, hence the natural choice of generative RNN controller models as a NAS search strategy.

However, training an RNN controller model via reinforcement learning necessarily requires significant amounts of data which is costly to generate where each data point is a candidate model trained a sufficient way to convergence. Further, as implemented by [Zoph et al, 2017], it is unclear how the learned features of a small-scale model, encoded in weights of a controller RNN, could be applied directly to scaling up to suit resource and performance objectives. The approach by [Zoph et al, 2018] to focus NAS on designing internal layer structure that can be repeated an arbitrary number of times addresses this limitation but does not address the task of describing the interaction between model depth and width as the model is scaled up.

Random-search exhibits strong performance, as noted by [Zoph et al, 2018]. This may be due to the tendency for random search to discover architectural configurations

from within discrete search spaces that principled search strategies typically do not as a result of preference for greedy exploitation of moderately successful architectural patterns. Random search is also expected to perform well, as illustrated by [Bergstra et al, 2012], in searching continuous spaces with significant imbalance in the importance of certain hyper-parameters relative to others. This leads to the second hypothesis for this project as follows.

Hypothesis 2: Further reductions in training time may be possible by applying Bayesian Optimisation in place of grid-search [Tan et al, 2020] or reinforcement learning [Tan et al, 2021] where a search strategy based on a Gaussian Process surrogate model is able to use efficiently sample from a continuous search space.

While the benefit of advancements by [Tan et al, 2020] and [Tan et al, 2021] are described by the authors in terms of training time, this seems to be an indirect benefit as training time is not included directly in the objective function, instead optimizing for a combination of accuracy, FLOPS [Tan et al, 2020], and model size [Tan et al, 2021]. This leads to the third hypothesis for this project as follows.

Hypothesis 3: Faster-to-train models can be more easily selected for by rewarding training time directly rather than FLOPS.

3.1 RESEARCH AIMS & OBJECTIVES

This project adopts the two step methodology of [Tan et al, 2020], and explores three key variations on the work by [Tan et al, 2020] and [Tan et al, 2021] as follows.

1. Application of Bayesian Optimization via SMBO of Gaussian Processes surrogate model as the NAS search strategy.
2. Global parameterisation, inspired by [Howard et al, 2017], of a family of models based on the structure of EfficientNetV2 as the search space.
3. Design of the global parameterisation scheme to:
    a. Use of continuous global parameters for maximising the advantage of flexible nonparametric Gaussian Processes models, and
    b. Enable re-use of global parameters in the procedure for scaling-up the model for increased performance.
4. Implementation of NAS objective function based on training time in lieu of FLOPS.

3.2 RESEARCH QUESTIONS

Research questions addressed by this project include:

1. Whether there is advantage to using Gaussian Processes in combination with a continuous global hyper-parameterisation scheme.
2. Whether the continuous global hyper-parameterisation scheme gives rise to scaled models which achieve commensurate improvement in performance.

3.3 RESEARCH/PROJECT SCOPE

The scope of this project includes the following tasks.

1. Design of a model generating procedure using continuous global hyper-parameterization to define the NAS search space (refer section 4.4.1).
2. Development of necessary functions for initializing, training, and evaluating candidate models.
3. Demonstration of baseline model(s) selection by NAS, applying Bayesian Optimisation using Gaussian Processes to guide search, and reporting of training and performance results.
4. Demonstration of scaling-up baseline model(s) by application of numeric hyper-parameters selected by NAS, and evaluation of scaled-up model(s) results.

The computational infrastructure available for this project outlined in Section 5 is permissive of distributed training. Optimisation of functions developed for initializing and training candidate models will include development for utilising distributed training.

4. METHODOLOGIES

4.1 Methods

Methods applied in this project are described as they relate to the following core aspects.

1. Data collection
2. Data analysis
3. Neural Architecture Search
4. Distributed training
5. Comparison with benchmarks
6. Standard training parameters

4.2 Data Collection

This work has focussed on the CIFAR100 image dataset for NAS and benchmarking, available for download using the torchvision python package. The reasons for selection of this dataset are as follows.

1. The dataset is designed for application to the task of image classification which was the focus task of the EfficientNet and EfficientNetV2 models by [Tan et al., 2020] and [Tan et al., 2021] respectively, and is the focus task for this project.
2. The dataset is of small enough scale (161MB on disk) to enable rapid iteration of training routines and the overall NAS procedure, compared with larger benchmark datasets such as ImageNet.
3. The image classes span a broad range of themes and contexts, which is important for development of model architectures that are able to generalise to a wide variety of domains.

The manageable scale of CIFAR100 comes at the expense of image resolution and number of images. CIFAR100 comprises 60,000 images of resolution 32x32 pixels pre-split into 50,000 training and 10,000 test images, with equal representation of each of 100 classes. The ImageNet dataset by comparison comprises 1,3M images of 1,000 classes at an average resolution of 469x387 pixels.

While CIFAR100 images can be scaled up to achieve higher model performance, this can be only the result of greater information encoded in the featuremap rather than greater degree of detail, specifically fine-grained detail, available at higher resolution images. Extension of this work could explore application of NAS to larger datasets

such as ImageNet or Open Images, in addition to other tasks such as object detection or semantic segmentation.

4.3 Data Analysis

A brief exploration was conducted to confirm the expected structure and content of the CIFAR100 dataset, and determine any necessary transformations to prepare images for training and testing. CIFAR100 is a well-structured and standardised benchmark dataset with all images of equal 32x32 resolution and 3-channels (no greyscale images). After transforming images to tensors, the pixel values are normalised from the range [0, 225] to the range [0, 1]. Analysis of these normalised images was conducted to determine channel-wise mean and standard deviation as follows.

Mean: (0.5071, 0.4866, 0.4409)

Standard deviation: (0.2009, 0.1984, 0.2023)

These statistics were then used to standardise images before training and inference.

4.4 Neural Architecture Search

This work aims to build upon the strategy developed by [Tan et al, 2020] for discovering sound model architecture principles which can be applied when scaling up to larger model sizes for higher performance. Where [Tan et al, 2020] uses separate mechanisms for NAS and scaling, (SMBO via controller RNN training for NAS and then compound scaling using a free parameter ϕ, refer Section 2.2), this project proposes to couple the NAS and scaling mechanisms by describing the search space in terms of continuous parameters that can then be directly applied to model scaling.

The main ideas explored in this project are:

1. To implement a continuous global parameterisation scheme suitable for the application of Gaussian Processes as the surrogate function for NAS search strategy.
2. To apply Bayesian Optimisation via SMBO of Gaussian Process surrogate models as a search strategy in place of grid search, reinforcement learning, or random search and,

3. For the continuous global parameterisation scheme to be directly applicable to scaling up the selected model architecture.

Methods applied in this work for approaching the three main considerations of NAS; search space, search strategy, and objective function, are described as follows.

4.4.1 Search space

4.4.1.1 Baseline model

The approach taken to designing the search space is to adopt certain architectural precepts from the EfficientNetV2 baseline model that were shown by [Tan et al, 2021] to be suitable for the dataset and objective. The search space is then designed to respect those precepts, and such that the baseline model is recovered from a point in that search space. The architecture of the smallest variant of the EfficientNetV2 model family, "EfficientNetV2-S" is shown in Figure 12.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

Figure 12 - EfficientNetV2-S architecture,
reproduced from [Tan et al, 2021] Table 4.

The architectural precepts observed from this baseline model and applied to the search space for this work are as follows.

1. Overall model composition takes the form of [head / body / classifier] with
    a. head consisting of Conv3x3,
    b. body consisting of Fused-MBConv and MBConv layers, and
    c. classifier consisting of Conv1x1 & Pooling & Linear layers.
2. Fused-MBConv used for early body layers, transitioning to MBConv for later body layers.
3. All Fused-MBConv and MBConv layers using 3x3 kernel convolutions.
4. Channel widths increase monotonically with model depth.
5. Spatial contraction layers distributed throughout the model body.

6. Residual connections and stochastic depth included at each layer with input / output of identical shape. Stochastic depth probability interpolated from 0 to 0.2 over the depth of the model.

The family of models searched in this work therefore conform to the schematic in Figure 13.

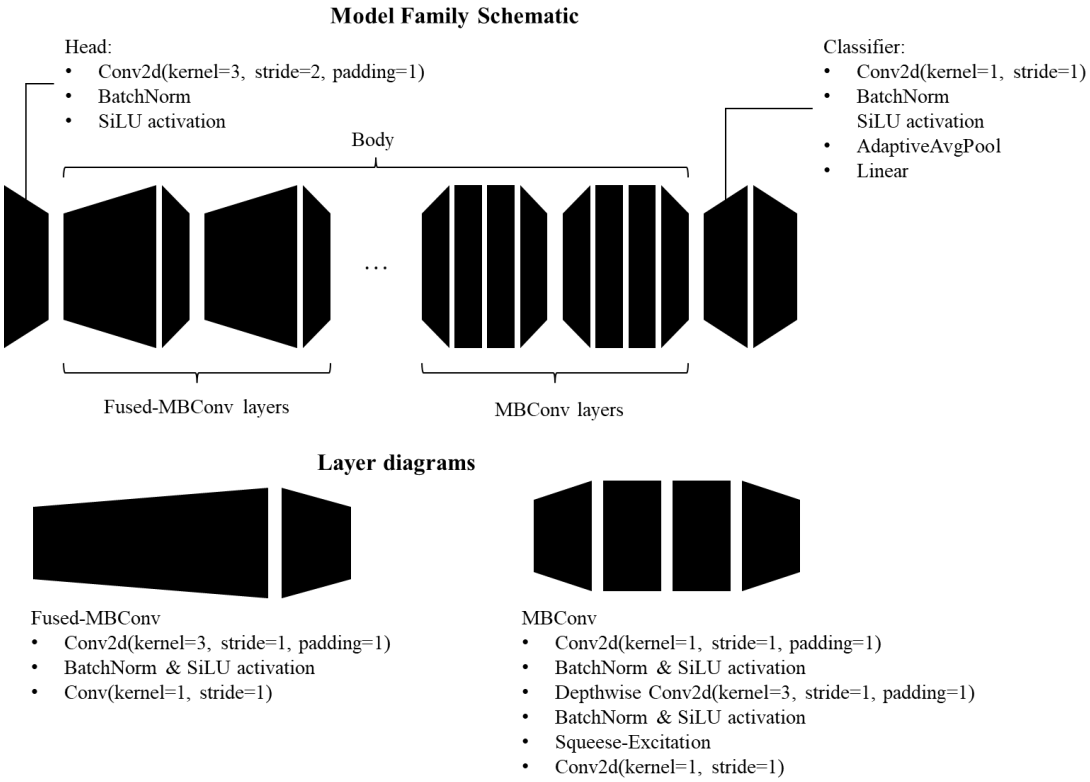


Figure 13 - Schematic for the family of architectures searched.

Much of the variation between models searched is within the body of the model, in particular:

1. Number of layers of each type (Fused-MBConv, MBConv),
2. Spatial resolution at each layer,
3. Number of input / output channels at each layer
4. Expansion ratio of each layer
5. Squeeze-Excitation bottleneck ratio at each MBConv layer

While there are many ways in which the above variation could be parameterised, two variations were developed for the purpose of this work, referred to as “BetaNet2”

and “BetaNet3” for their respective use of 2 or 3 distinct beta distributions in the procedure for constructing models.

4.4.1.2 Search space parameters

The search space is described by ranges of continuous parameters that are used in the procedure for constructing BetaNet2 and BetaNet3 models. In both cases, the model scale is primarily based on the input spatial resolution. The model is then constructed using the continuous parameters described in Table 1, which together describe the kernel of self-similarity for each model architecture. All parameters itemised are common to both the BetaNet2 and BetaNet3 search spaces unless stated otherwise.

Table 1 - Parameters used for construction of BetaNet model

Parameter(s)	**Searched**	**Purpose**
R	64, 224	Input image resolution, used as model scaling parameter.
k	[2, 10]	log2 of the maximum permitted spatial resolution in final model layers
K	[2, 4]	Determines total number of layers
Da, Db	[-0.5, 0.5]	Determines spatial contraction layer spacing
Wa, Wb	Wa: [0, 0.5] Wb: [-0.5, 0]	Determines interpolation from 3 to final channel width as a function of model depth for **BetaNet2** only.
WLa, WLb	[-0.5, 0.5]	Determines channel width adjustment layer spacing for **BetaNet3** only.
WCa, WCb	[-0.5, 0.5]	Determines interpolation from 3 to final channel width as a function of model depth for **BetaNet3** only.
G0	[2, 5]	Determines final channel width
G1	[1, 3]	Determines latent vector size
B	[0, 1]	Model depth at which to transition from Fused-MBConv to MBConv layers.
E0, E1	[2, 4]	Fused-MBConv and MBConv layer expansion ratios interpolated from E0 to E1 over model depth
S0, S1	[10, 20]	Squeeze-Excitation layer (MBConv layers) bottleneck ratios interpolated from S0 to S1 over model depth

The BetaNet2 and BetaNet3 models are constructed using the parameters itemised in Table 1 according to the procedure as follows.

a. Common to BetaNet2 and BetaNet3:

- Select an input image resolution **R**.
- Determine the number of 3-kernel, 1-padded, 2-stride convolutions L necessary to reduce the input spatial resolution from R to less than $2^{\mathbf{k}}$ pixels. The final layer pools over the remaining $<2^{\mathbf{k}}$-dimensional spatial resolution.
- Multiply parameter **K** by L to determine the total model layers.
- Apply one 3-kernel, 1-padded, 2-stride convolutional layer with batch normalisation and SiLU activation as the model "head".
- Assign remaining spatial contraction layers by means of parameters **Da** and **Db**, and the procedure outlined in Section 4.4.1.3 (Figure 15, 16).
- Multiply parameter **G0** by R to determine the channel width of the final spatially extended layer.

BetaNet2 only:

- Determine the number of input / output channels for each layer by means of parameters **Wa** and **Wb**, and the procedure outlined in Section 4.4.1.3 (Figure 15).

BetaNet3 only:

- Assign channel width adjustment layers by means of parameters **WLa** and **WLb**, and the procedure outlined in Section 4.4.1.3 (Figure 16, middle).
- Determine the number of input / output channels for each layer according to parameters **WCa** and **WCb** describing channel width as a function of model depth.

b. Common to BetaNet2 and BetaNet3:

- Multiply **G1** by the channel width of the final spatially extended layer to determine the size of the classifier latent space vector.
- Determine Fused-MBConv and MBConv layers by means of parameter **B**.
- For Fused-MBConv and MBConv layers, apply expansion ratios interpolated from **E0** to **E1** over the depth of the model.

- For MBConv layers, apply Squeeze-Excitation layer bottleneck ratios interpolated from **S0** to **S1** over the depth of the model.
- For non-spatial resolution reduction layers, apply stochastic depth by applying **SDP0** and **SDP1** interpolated over the depth of the model.
- Apply classifier consisting of Conv1x1, Average Pooling, and Linear layers.

This procedure is reasonably fast (less than 1 second to construct selected BetaNet2 and BetaNet3 models with input resolution 224x224, refer Section 7.2), contributing an acceptably small amount of time to the overall NAS procedure.

4.4.1.3 Model shape parameters

Procedures were developed for parameterising the shape of the body of the BetaNet2 and BetaNet3 models in terms of the spatial resolution and channel width as a function of the depth of the model. These procedures make use of the Beta distribution for assigning spatial contraction and channel width adjustment layers, and for interpolating between the number of model input and output channels to determine the model output channels as a function of the depth of the model.

The Beta distribution is a continuous probability distribution determined by two parameters and defined on the interval [0, 1] which can be used to determine the spacings between spatial reduction layers by means of the percent point function. Beta distribution percent point functions for a range of parameterisations are illustrated in Figure 14. The cumulative Beta distribution function (or percent point function) was chosen for the following reasons:

1. It is highly expressive, with the ability to approximate the shape of linear, polynomial, logarithmic, sigmoidal, or exponential functions at suitable parameterisation (refer Figure 14).
2. It is defined on the unit domain, which can be readily mapped to arbitrary model depth or desired domain.
3. It requires only two parameters to determine the distribution shape, and these parameters can be transformed by exponentiation (refer Figure 14) to make the parameter ranges linear and thus easier for modelling by Gaussian Processes.

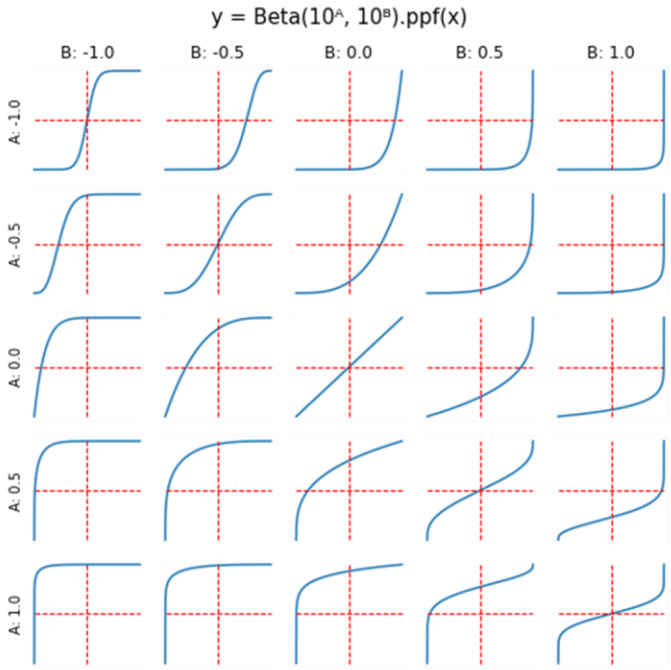


Figure 14 - Beta distribution percent point functions for a range of parameterisations. The greatest variance in Beta distribution percent point function shape is apparent within the parameter range $[10^{-1}, 10^{1}]$.

The procedures for assigning spatial contraction layers, width adjustment layers, and layer output channels for BetaNet2 and BetaNet3 are illustrated in Figure 15 and Figure 16 respectively and described as follows.

c. Common to BetaNet2 and BetaNet3:

To select the spatial reduction layers, **L** values are generated equally spaced on the interval [0, 1]. These values are transformed by means of a suitably parameterised Beta distribution percent point function as shown in Figure 15 and Figure 16 (left), and the corresponding model layers are chosen as those nearest matching the transformed values at percentage depth of the model.

BetaNet2 only:

To find the number of output channels for a layer, log_2 of the spatial resolution of the layer is mapped to the interval [0, 1], and transformed according to a suitably parameterised Beta distribution, before being mapped to a number of output channels on the interval [3, **G0** x **R**] corresponding to the model input channels and the final channel width generated according to the procedure outlined in Section 4.4.1.2.

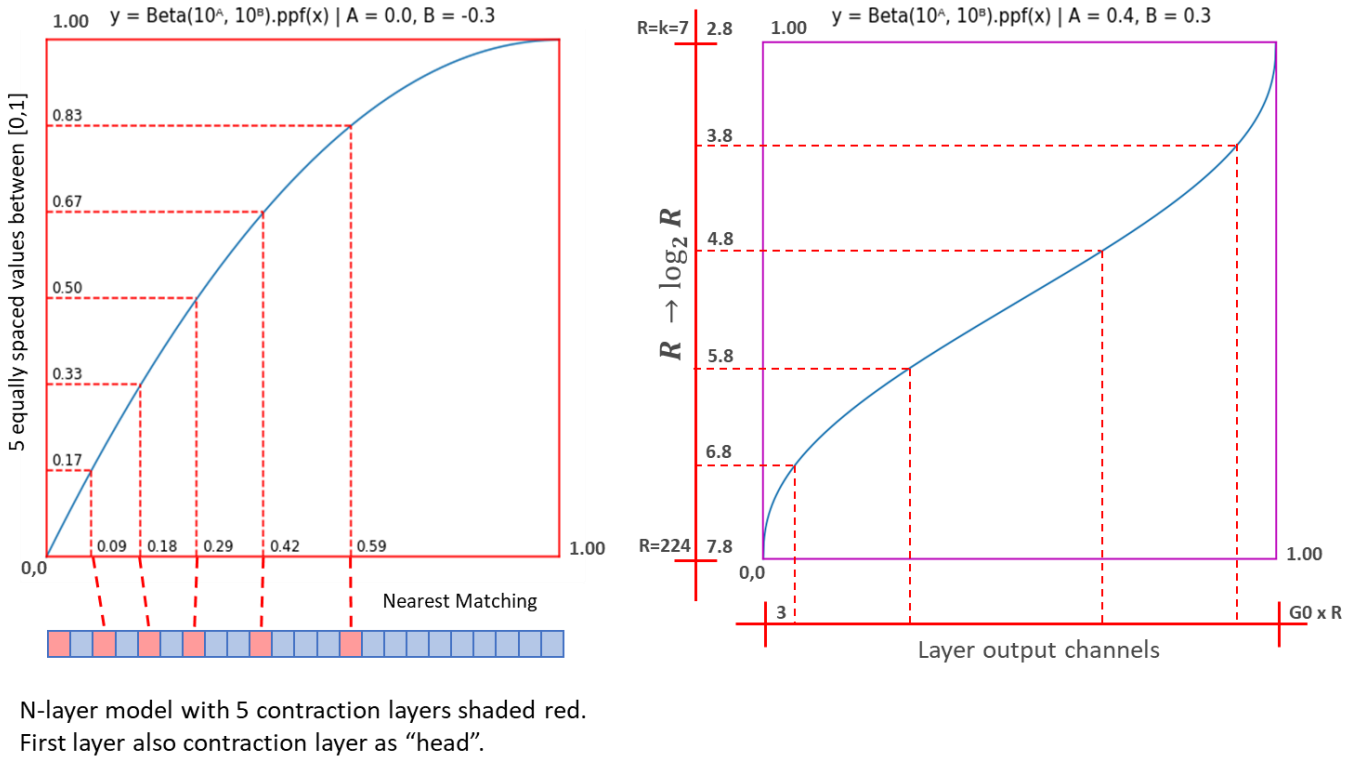


Figure 15 - BetaNet2 procedure for determining spatial contraction layers and layer output channels using 2 Beta Distributions.

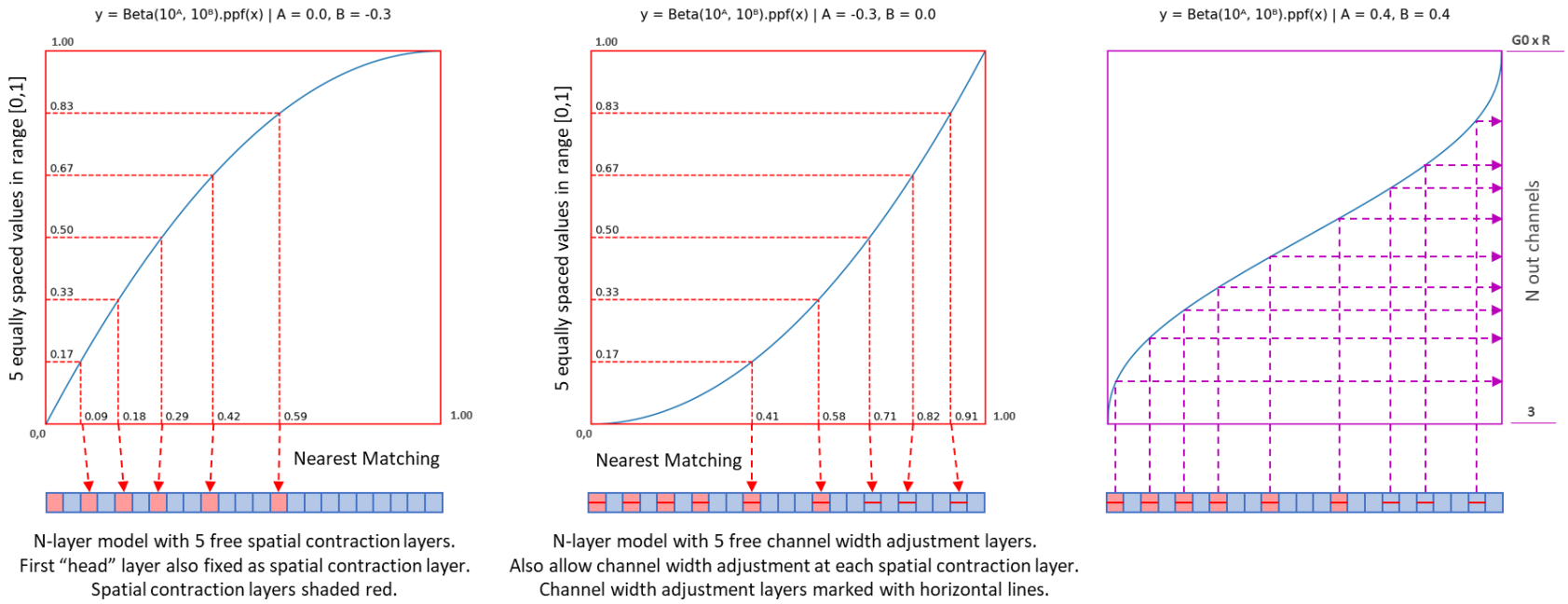


Figure 16 - BetaNet3 procedure for determining spatial contraction and channel width adjustment layers, as well as layer output channels using 3 Beta Distributions.

BetaNet3 only:

An additional **L** channel width adjustment layer placements are assigned by generating L equally spaced values are generated on the interval [0, 1]. These points are transformed by means of a suitably parameterised Beta distribution percent point function as shown in Figure 16 (middle), and the corresponding model layers are chosen as those nearest matching the transformed values at percentage depth of the model. Channel width adjustment is also performed at spatial contraction layers. Channel width adjustment layers coinciding with spatial contraction layers are not replaced; in these cases there are simply fewer than **L** dedicated channel width adjustment layers.

Channel widths at the adjustment layers are set according to a suitably parameterised Beta distribution as shown in Figure 16 (right) where the percentage depth of the model for the layer is transformed according to the Beta distribution, and the resulting values mapped to a number of output channels on the interval [3, **G0** x **R**] corresponding to the model input channels and the final channel width generated according to the procedure outlined in Section 4.4.1.2.

4.4.1.4 BetaNet2 and BetaNet3 variation rationale

The two variations tested in this work, BetaNet2 and BetaNet3, explore a trade-off between model expressivity and search space complexity. BetaNet2 is a simpler search space with fewer degrees of freedom (involving 4 Beta distribution parameters, compared with 6 Beta distribution parameters for BetaNet3) and therefore permitting less expressivity compared with BetaNet3.

The reduced expressivity of BetaNet2, however, comes with the advantage of a simpler search space which the search strategy described in Section 4.4.2 is better able to search effectively. In addition BetaNet2 requires in principle fewer layers with different shape input and output featuremap, and therefore can retain residual connections and stochastic depth around more of the layers of the network. This should have the effect of stabilising training as shown by [Kaiming et al, 2015], and imposing stronger regularisation which should advantage deeper networks.

4.4.2 Search strategy

NAS is implemented using the open-source Adaptive Experimentation Platform "Ax" from Facebook/Meta Research. The search space is first described in terms of parameters and associated ranges to search. A surrogate model (Gaussian Process) is used to approximate the objective function which is the evaluation of candidate models on a suitable proxy task.

The search strategy is to apply Bayesian Optimisation via SMBO of a Gaussian Process model. Overall, SMBO proceeds as follows.

1. 100 models are sampled from the search space in the form of hyper-parameter vectors according to a "Sobol sequence". A Sobol sequence is a quasi-random sequence that distributes more uniformly throughout a space compared with random sampling. Models are evaluated according to the objective function (refer Section 4.4.3) which serve as the seed for the next phase of the NAS procedure.
2. The Gaussian Process model is used to approximate the objective function given the sampled model hyper-parameters and evaluations. The Gaussian Process model describes the search space in terms of the expectation and uncertainty of the objective function.
3. An acquisition function is used to select the next point within the search space to sample. The Gaussian Process Expected Improvement (GPEI) is used in this project as the acquisition function which samples points from the search space where the expectation and uncertainty of the Gaussian Process model suggests it is plausible that the objective function may return better results. The selected point and corresponding model are evaluated according to the objective function.
4. Steps 2 and 3 are repeated until 200 trials have been evaluated with non-trivial results (refer Section 4.4.3).

4.4.3 Objective function

Model evaluation is with respect to a proxy task defined as top-1 validation accuracy achieved within 5 minutes of training time at an input image resolution of 224x224 pixels. Imposition of a training time budget (Hypothesis 3) has the effect of preferring models which achieve higher performance in less time.

Due to the wide variety of model architectures encompassed by the search spaces described in Section 4.4.1, many valid model descriptions resulted in models of impractical size; either with an infeasible number of parameters, or with a feature map size too large to fit into memory. It was not, however, immediately obvious which hyper-parameter ranges would give rise to either of these undesirable conditions, and indeed the feasibility or otherwise of one parameter range was strongly conditional on the other parameter ranges.

Therefore, instead of imposing prior constraints on the search space in pursuit of desirable scale properties, model hyper-parameters were sampled and tested. Models with a number of parameters greater than 30M, or which generated a feature map that would not fit into memory, returned an evaluation of zero. These conditions are referred to as "trivial" results. As described in Section 4.4.2, 100 models were sampled according to a Sobol sequence within the search space, where 100 was selected empirically to obtain 20-30 non-trivial results during this initial phase of the NAS. The GPEI phases of the NAS would then proceed until 200 non-trivial evaluations had been obtained in total.

4.5 Distributed training

The advantage of distributed training was initially demonstrated by [Krizhevsky et al, 2012] in development of AlexNet, and is now widely employed to reduce training time for deep learning models.

A distributed training routine was developed for this project using the Pytorch `torch.nn.parallel.DistributedDataParallel` module which integrates with a Pytorch model training routine to aggregate and synchronise gradients computed on each GPU in the cluster during the backward pass. Each available GPU receives a copy of the model with identical weights, and a subset of the dataset, disjoint from the subsets allocated to each other GPU. Each GPU then carries out a forward pass, loss calculation, and backward pass. During the backward pass, each GPU collects and aggregates the gradients computed by every other GPU in the cluster. Each GPU then updates its model weights according to the now synchronised gradients.

This procedure is advantageous in the regime where the time for each GPU to compute the forward and backward pass on a batch of data is much greater than the time required to synchronise gradients between GPUs, which is necessarily sensitive

to the model, dataset, and inter-GPU communication pathway. Per-epoch training time for a ResNet50 model trained on the CIFAR-100 dataset utilizing between 1 and 6 GPUs is shown in Figure 17.

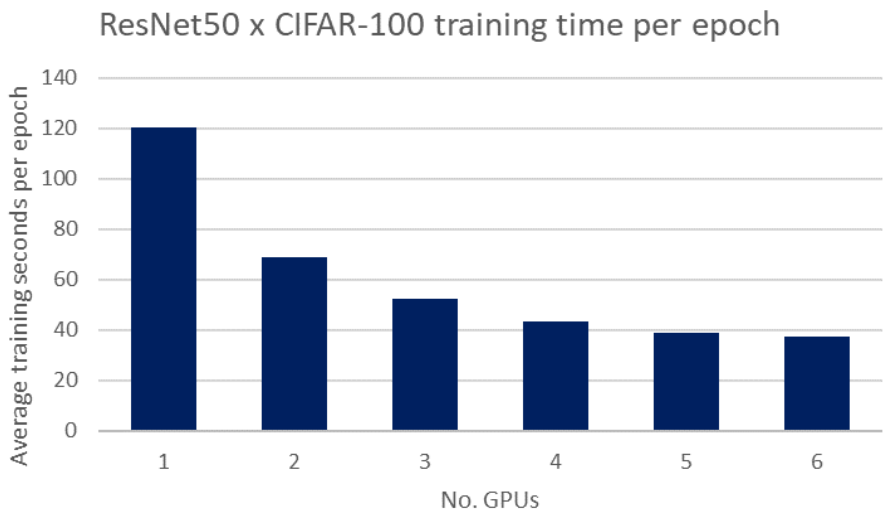


Figure 17 - Per-epoch training time for ResNet50 model trained on CIFAR-100 dataset utilising between 1 and 6 GPUs.

Gradient compression was employed to further accelerate training, whereby the gradients at each GPU are first downcast to 16-bit floats before they are synchronised to reduce the inter-GPU communication time. This also has a slight regularisation effect.

To support rapid iteration with the convenience of a Jupyter notebook environment, the distributed training routine was developed as a python script which is called from within the notebook environment and provided access to the notebook environment namespace where model hyper-parameters are defined.

4.6 Comparison with benchmarks

Benchmark models were selected to compare results of the BetaNet2 and BetaNet3 NAS. Benchmark models include those structurally related to the BetaNet components including MobileNetV2, EfficientNet-B0, and EfficientNetV2-S, as well as ResNet50 which is another commonly used benchmark. Summary details for these benchmark models are given in Table 7 and Table 8.

Whereas ResNet50, MobileNetV2, and EfficientNet-B0 benchmark models assume an input resolution of 224x224 pixels, EfficientNetV2-S assumes an input of

384x384 pixels. As a result, BetaNet2 and BetaNet3 models were generated at R=224 and also scaled up to R=384 for comparison with these two groups of benchmark models respectively, testing the scalability of the discovered BetaNet2 and BetaNet3 model architectures (Hypothesis 1).

4.7 Standard training parameters

A training routine was developed which achieves reasonable performance for benchmark models. Features of this training routine are itemised in Table 2.

Table 2 - Benchmark training routine hyper-parameters

Feature	**Description**
Parallel GPUs	6
Batch size	100 (effective batch size 600)
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)
Learning rate	1e-3
Data augmentation	• Torchvision Auto-Augment (CIFAR-10 policy) • Random Horizontal Flip • Random Resized Crop
Learning rate scheduling	Reduce Learning Rate On Plateau (patience = 5 epochs)

These settings were not specifically tuned to any one benchmark model, but developed to give reasonable performance on all benchmark models, and is also used for the BetaNet2 and BetaNet3 NAS routine.

5. RESOURCES

5.1 Hardware

Computational resources made available by Strong Compute for this work include a compute cluster with the following attributes.

Table 3 - Infrastructure and available resources

Component	Type
CPU	16 core AMD
Random Access Memory	128 GB
Storage	2 TB
GPU	6 x NVIDIA (A10 Equivalent)

5.2 Software

Software and versions available to be used for this work include the following.

1. Python (3.8.13)
    a. numpy (1.22.4)
    b. pandas (1.4.3)
    c. sklearn (0.24.2)
    d. matplotlib (3.5.2)
    e. JupyterLab (2.3.2)
2. Pytorch (1.13.0a0+d321be6)
    a. torchvision (0.14.0a0)
3. Ax (Bayesian Optimization) (0.2.6)

6. MILESTONES / SCHEDULE

The overall timeline to which this work was completed is shown below. Refer to Appendix B for bi-weekly progress reports.

Table 4 - Project timeline

Milestone	Tasks	Date
Week-1	● Identify work to expand upon.	4 August
Week-2	● Architecture exploration and domain-focused research.	11 August
Week-3	● Draft scope proposal. ● Selection of dataset.	18 August
Week-4	● Scope refinement and detail. ● Progress with development of prototype model code.	25 August
Week-5	● Finalise and submit Proposal Report	4 September
Week-6	● Complete development environment set-up ● Obtain and prepare ImageNet 2012 dataset ● Obtain and prepare CIFAR-100 dataset ● Implement NAS with Bayesian Optimisation	15 September
Week-7	● Develop implementation of distributed training to accelerate progress ● Establish baseline training routine and benchmark model results	22 September
Week-8	● Obtain results from Bayesian Optimisation NAS ● Refine hyper-parameter ranges to search ● Refine NAS objective function to include desirable scaling properties ● Prepare draft progress report	29 September
Week-9	● Finalise and submit Progress Report	9 October
Week-10	● Obtain further results from NAS with Bayesian Optimisation ● Obtain results from scaling model kernel selected ● Commence assembly of results for project presentation ● Commence draft Project Report	20 October
Week-11	● Final round of iteration on NAS with Bayesian Optimisation and obtaining results from scaled model ● Complete project presentation	27 October
Week-12	● Submit Project Presentation ● Complete Project Report	13 November
Week-13	● Finalise and submit Project Report	20 November

7. RESULTS

7.1 BetaNet2 and BetaNet3 NAS

7.1.1 BetaNet2

The objective function evaluation of candidate models trailed throughout the course of the BetaNet2 NAS is shown in Figure 18. An initial 100 Sobol samples are evaluated, the majority of which return trivial zero evaluation due to model or feature map size. This is followed by steady improvement in the model evaluations as GPEI narrows the focus of sampling to regions of the search space with higher performance.

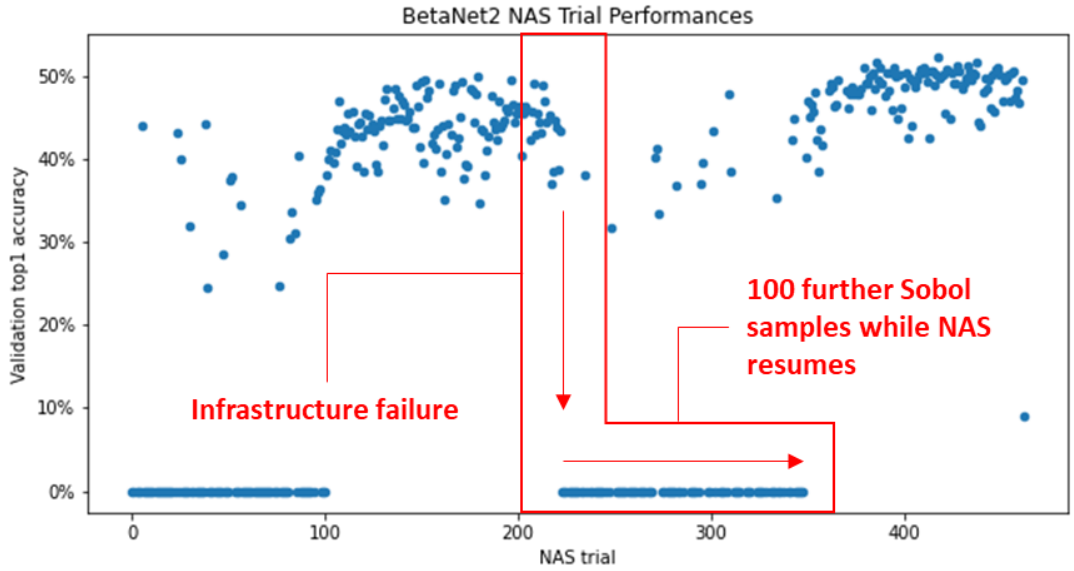


Figure 18 - Top-1 validation accuracy of BetaNet2 candidate models. An infrastructure failure occurred at trial 215, resulting in the need to restart the NAS procedure including collection on an additional 100 Sobol samples.

Infrastructure stability was an inhibiting factor for this project in that GPU drivers had the propensity to fail unpredictably. One such failure is evident from the BetaNet2 NAS results in Figure 18, where the evaluations plummet at trial 215. After the infrastructure was reinstated, the experiment was started again with a second round of Sobol sampling, following which GPEI sampling resumed. Results saved prior to the failure were recovered and injected into the experiment database to inform resumed GPEI sampling.

Candidate model evaluations are shown in Figure 19. The top 50 candidate models superimposed on top of the top 100 candidate models demonstrates the

convergence of the BetaNet2 NAS toward a maximal region of the architecture search space.

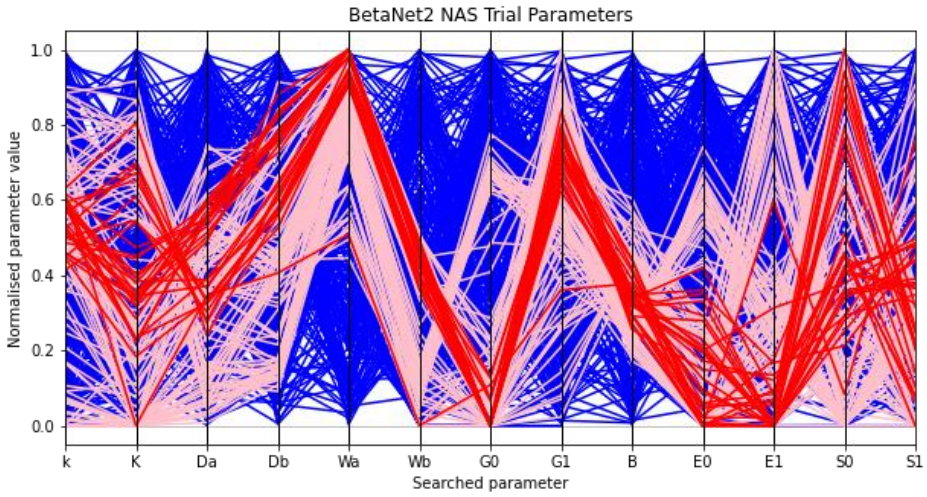


Figure 19 - BetaNet2 NAS highest performing 50 (red), 100 (pale red), and other (blue) candidate models. Spanning lines represent candidate models where the point of intersection with the vertical axis represents the normalised value for that hyperparameter.

These results also indicate the varying degrees of uncertainty of the GP model with respect to each hyper-parameter value. The GP model for example appears more confident in the optimal value for parameters **Wa**, **Wb**, **G0**, and **G1** as the top 50 models have much lower variance in these normalized parameter values compared with hyper-parameters such as **K**, **Da**, **Db**, **S0**, and **S1**.

As with other NAS search spaces and strategies (such as RNN controller models), this approach generates a dataset of evaluated model architectures. The technique demonstrated in this project, however, has the advantage that similarity between candidate architectures can be readily quantified due to the exclusive use of continuous global hyper-parameters.

The top 30 BetaNet3 candidate models are represented in Figure 20, hierarchically clustered on their standardised hyper-parameter values. Clusters correspond to regions in the vicinity of local maxima in the architecture search space, each of which may demonstrate different scaling and performance properties.

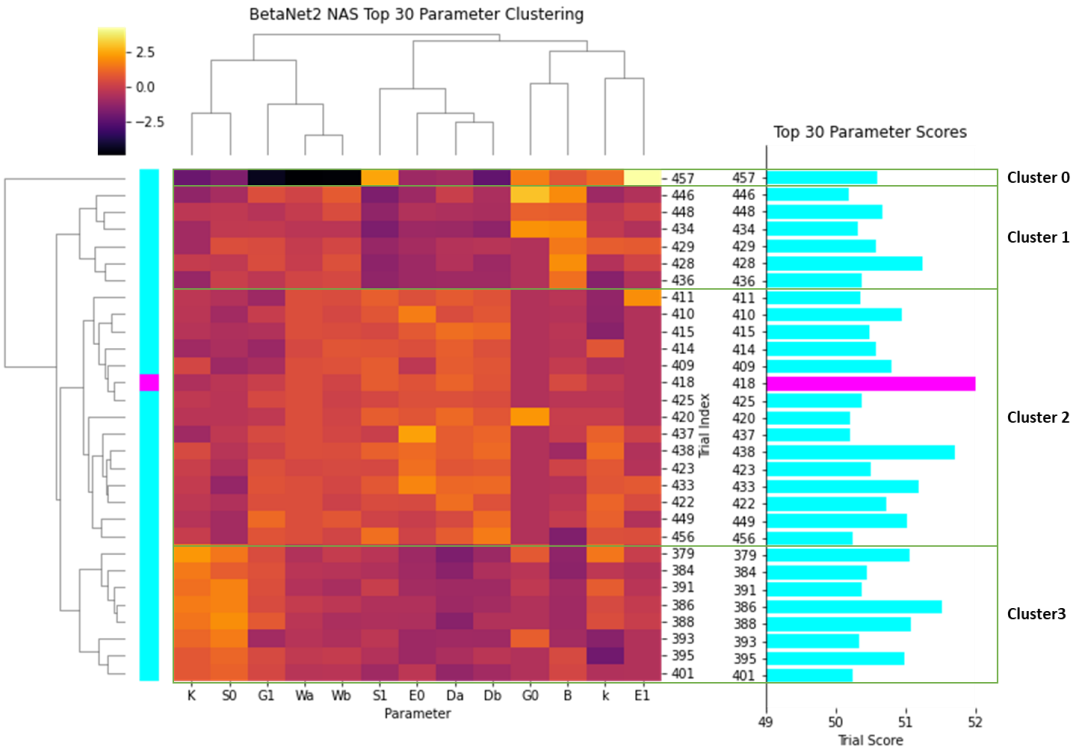


Figure 20 - BetaNet2 NAS highest performing 30 candidate models hierarchically clustered by standardised hyper-parameter values. Each cluster corresponds to a local maxima in the architecture search space. Cluster 2 includes the highest performing candidate model (purple).

The GP model, fit to all of the candidate model evaluations, returns a single set of global hyper-parameters which corresponds to the global maxima of the expectation surface, or the expected best performing model. These "best" model hyper-parameters for BetaNet2 are shown in comparison with each of the four top-30 clusters mean hyper-parameter values and mean cluster evaluations in Table 5.

Table 5 - BetaNet2 best and top 30 4-cluster mean hyper-parameter values

Cluster	Best	Cluster 0	Cluster 1	Cluster 2	Cluster 3
Mean eval.	N/A	50.59	50.56	50.78	50.75
k	3.23	3.23	3.03	3.09	3.06
K	3.95	3.00	3.51	3.69	4.30
Da	0.05	-0.15	-0.12	0.05	-0.19
Db	0.33	-0.09	0.08	0.29	0.09

Cluster	Best	Cluster 0	Cluster 1	Cluster 2	Cluster 3
Wa	0.50	0.16	0.47	0.50	0.44
Wb	-0.03	-0.50	-0.04	-0.04	-0.11
G0	2.00	2.22	2.12	2.02	2.04
G1	2.48	1.72	2.48	2.47	2.52
B	0.29	0.38	0.42	0.32	0.30
E0	3.12	2.00	2.06	2.92	2.11
E1	2.00	4.39	2.24	2.22	2.10
S0	18.14	11.67	19.41	16.95	27.34
S1	19.93	25.17	11.08	18.67	14.52

The best hyper-parameter values and four top-30 clusters normalised mean hyper-parameter values for BetaNet2 are shown in Figure 21. Cluster 0 exhibits significant variation from Clusters 1, 2, and 3. Cluster 0 however, comprises only a single model evaluation and therefore its strong mean evaluation may be less reliable.

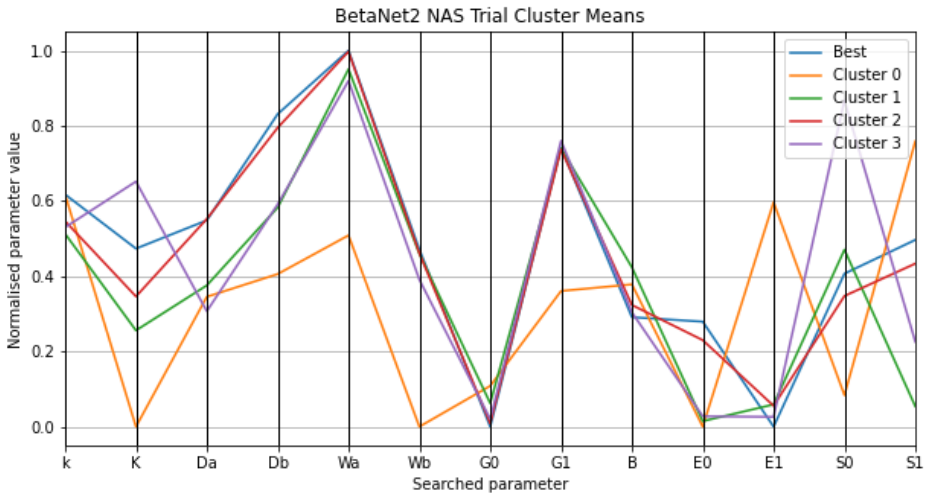


Figure 21 - BetaNet2 NAS top 30 candidates 4-cluster mean hyper-parameter values. The best (recommended) hyper-parameter values in blue closely match Cluster 2.

7.1.2 BetaNet3

BetaNet3 candidate evaluations are shown in Figure 22 and Figure 23. No infrastructure failure occurred during this experiment, and so after the 100 initial Sobol samples, the search converges continuously toward higher performing models.

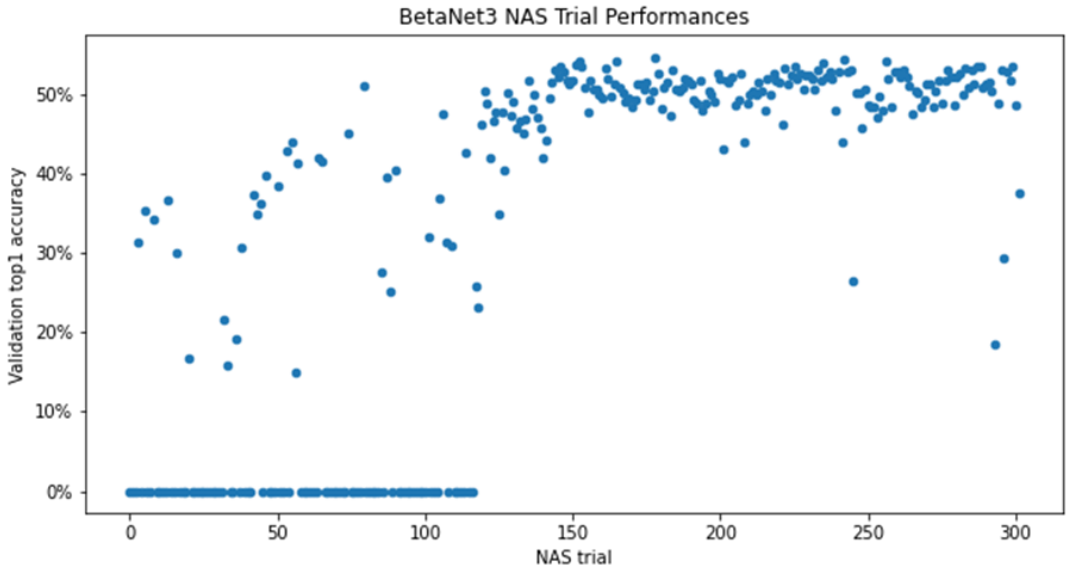


Figure 22 - Top-1 validation accuracy of BetaNet3 candidate models. 100 initial Sobol samples with high rejection rate (returning evaluation of 0) succeeded by GPEI sampled models until 200 non-rejected models have been evaluated.

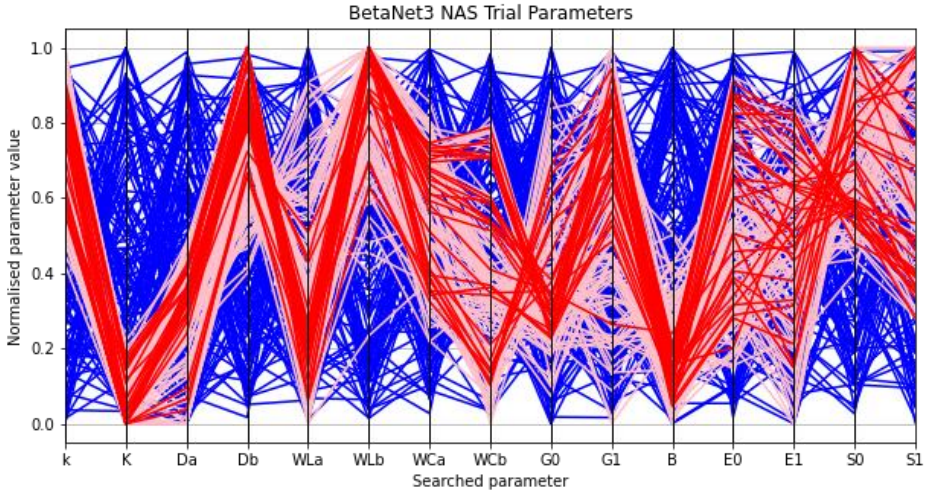


Figure 23 - BetaNet3 NAS highest performing 50 (red), 100 (pale red), and other (blue) candidate models. Spanning lines represent candidate models where the point of intersection with the vertical axis represents the normalised value for that hyperparameter.

The BetaNet3 GP model exhibits higher confidence in the optimal value for parameters **k**, **K**, **Db**, and **B** as the top 50 models have much lower variance in these

normalized parameter values compared with hyper-parameters such as **WCa**, **WCb**, **E0**, **E1**, **S0** and **S1**.

The top 30 BetaNet3 candidate models are represented in Figure 24, hierarchically clustered on their standardised hyper-parameter values. Clusters correspond to regions in the vicinity of local maxima in the architecture search space, each of which may demonstrate different scaling and performance properties.

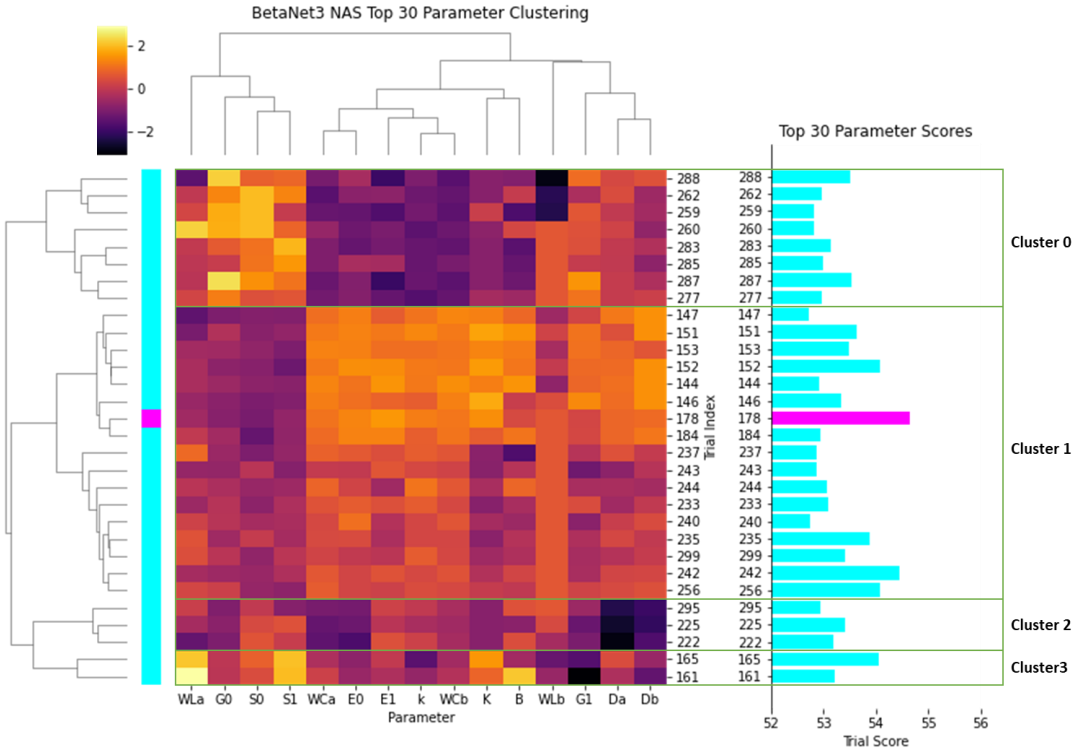


Figure 24 - BetaNet3 NAS highest performing 30 candidate models hierarchically clustered by standardised hyper-parameter values. Cluster 1 includes the highest performing candidate model (purple).

Best model hyper-parameters for BetaNet3 are shown in comparison with each of the four top-30 clusters mean hyper-parameter values and mean evaluations in Table 6. Mean evaluations for BetaNet3 indicate stronger performance than BetaNet2 clusters. The best hyper-parameter values and four top-30 clusters normalised mean hyper-parameter values for BetaNet3 are shown in Figure 25.

From Table 5 and Table 6, Both BetaNet2 and BetaNet3 models prefer a depth multiplier K of near to 3, the minimum value searched. This may represent a bias toward faster training models as an artifact of the objective function design, and therefore K could be used to scale model depth manually for fine-tuning on a scaled-up task.

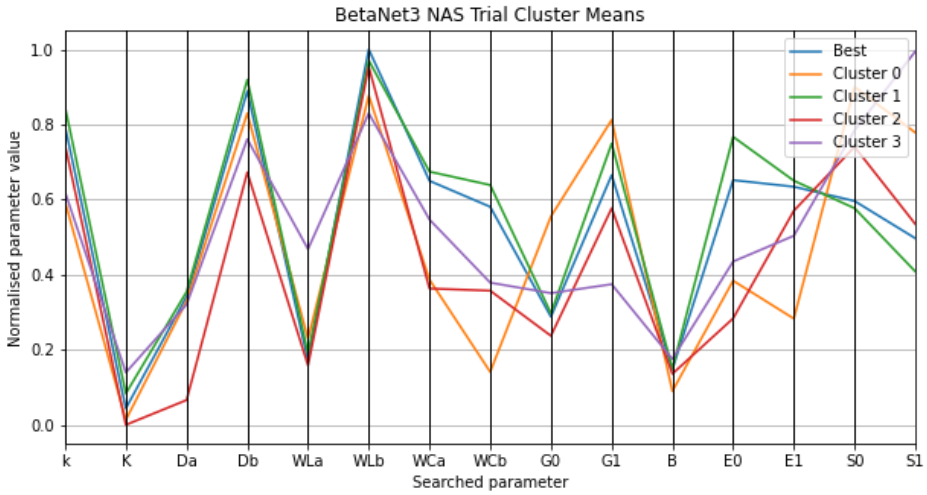


Figure 25 - BetaNet3 NAS top 30 candidates 4-cluster mean hyper-parameter values. The best (recommended) hyper-parameter values in blue closely match Cluster 1.

Table 6 - BetaNet3 best and top 30 4-cluster mean hyper-parameter values

Cluster	Best	Cluster 0	Cluster 1	Cluster 2	Cluster 3
Mean eval.	N/A	53.09	53.42	53.17	53.64
k	3.59	3.19	3.69	3.49	3.24
K	3.09	3.03	3.17	3.00	3.28
Da	-0.16	-0.17	-0.15	-0.43	-0.18
Db	0.39	0.33	0.42	0.17	0.26
WLa	-0.33	-0.27	-0.31	-0.34	-0.03
WLb	0.50	0.38	0.47	0.45	0.33
WCa	0.15	-0.11	0.17	-0.14	0.05
WCb	0.08	-0.36	0.14	-0.14	-0.12
G0	2.58	3.11	2.59	2.47	2.70
G1	2.33	2.62	2.50	2.15	1.75

Cluster	Best	Cluster 0	Cluster 1	Cluster 2	Cluster 3
B	0.15	0.09	0.15	0.14	0.18
E0	4.61	3.54	5.07	3.13	3.74
E1	4.54	3.13	4.60	4.28	4.01
S0	21.92	28.00	21.54	24.81	25.69
S1	19.94	25.57	18.17	20.70	29.90

Unlike the EfficientNetV2 backbone which included expansion ratios ranging from 4 at early Fused-MBConv layers to 6 at the final MBConv layers, BetaNet2 models preferred much smaller expansion ratios reducing from 3.12 at the first Fused-MBConv layer to 2 at the final MBConv layer. BetaNet3 preferred an approximately constant modest expansion ratio range from 4.61 for 4.54.

BetaNet2 implements transition from Fused-MBConv to MBConv layers at 0.29 of the depth of the overall model, whereas BetaNet3 appears to prefer to minimise the use of Fused-MBConv altogether, transitioning to MBConv at a depth of only 0.15 which from Table A3 and Table A4 (Appendix A) permits only one Fused-MBConv layer.

7.2 BetaNet2 and BetaNet3 performance and benchmark comparison

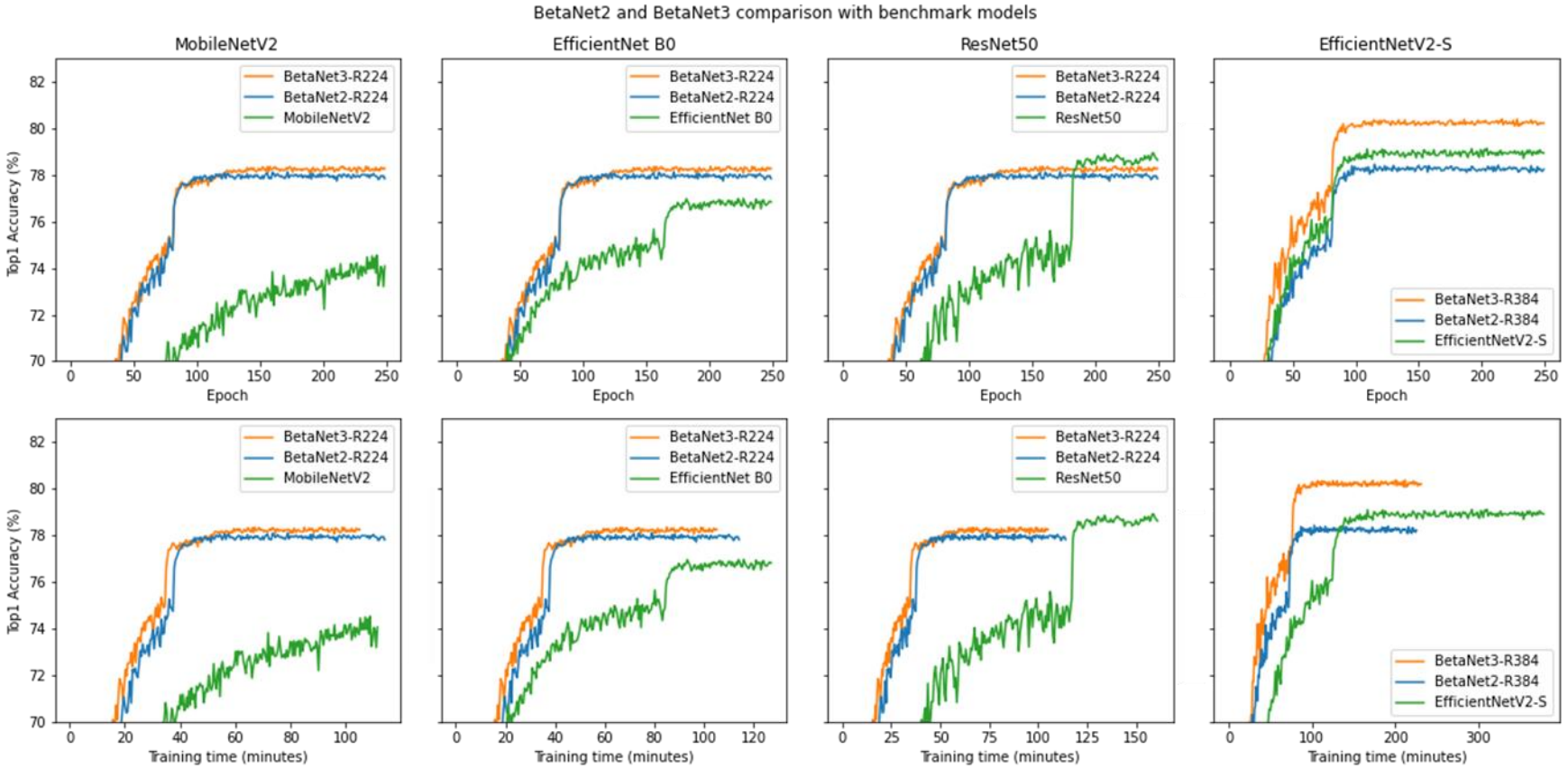


Figure 26 - Selected BetaNet2 and BetaNet3 models training histories compared with benchmark models. Top row presents top-1 test accuracy at each training set epoch, bottom row presents top-1 test accuracy as a function of time. Leftmost three columns compare BetaNet2 and BetaNet3 at R=224 with benchmark models with native input resolution of R=224. Rightmost column compares BetaNet2 and BetaNet3 scaled up to R=384 for comparison with the EfficientNetV2-S benchmark model with native input resolution of R=384. BetaNet2 and BetaNet3 models generally train faster than comparable benchmark models and achieve higher or comparable performance. BetaNet3 exhibits significant performance improvement at R=384, surpassing EfficientNetV2-S.

The training histories shown in Figure 25 compare the selected BetaNet2 and BetaNet3 model performance and training speed with benchmark models. The BetaNet2 and BetaNet3 models consistently train faster than benchmark models, evidence in favour of Hypothesis 3 (Section 3) that inference and therefore training speed can be selected for directly, returning models that are faster to train even than benchmark models for which this was a key objective such as MobileNetV2 and EfficientNet.

Presented in Table 7 and 8 are the key statistics and performance for comparison of BetaNet2 and BetaNet3 with benchmark models. BetaNet3 demonstrates significant performance improvement when scaled from the baseline R=224 up to R=384 for comparison with EfficientNetV2-S, surpassing the top-1 test set accuracy of EfficientNetV2-S. In contrast, BetaNet2 exhibits only slight performance improvement after scaling to R=384.

Table 7 - BetaNet2 and BetaNet3 benchmark comparison at R=224

Model	**No. params (M)**	**Top1 test accuracy (%)**	**Training time (hours)**
ResNet50	23.71	**78.91**	2.68
MobileNetV2	2.35	74.55	1.86
EfficientNet B0	4.14	76.97	2.11
BetaNet2-R224	8.86	78.11	1.90
BetaNet3-R224	**5.32**	78.34	**1.75**

Table 8 - BetaNet2 and BetaNet3 benchmark comparison at R=384

Model	**No. params (M)**	**Top1 test accuracy (%)**	**Training time (hours)**
EfficientNetV2-S	20.31	79.11	6.31
BetaNet2-R384	30.01	78.43	3.76
BetaNet3-R384	**15.00**	**80.37**	**3.86**

Refer to Appendix A for details of the architectures of the selected BetaNet2 and BetaNet3 models corresponding to the best global hyper-parameters presented in Table 5 and Table 6.

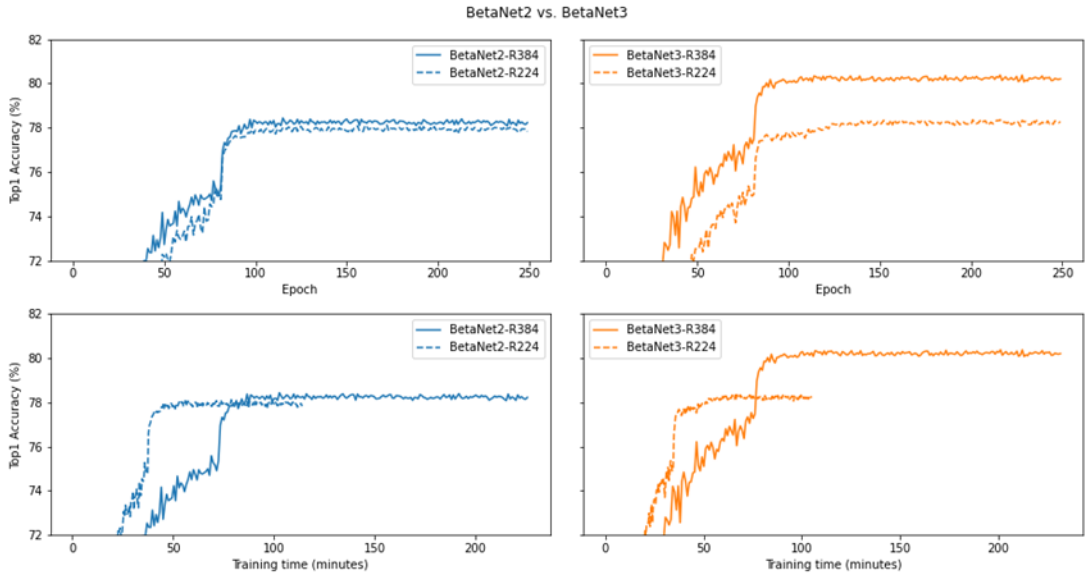


Figure 27 - BetaNet2 and BetaNet3 self-comparison at R=224 (before scaling) and R=384 (after scaling). Top row presents top-1 test accuracy with each training epoch, bottom row presents top-1 training accuracy as a function of minutes of training. BetaNet3 demonstrates stronger performance uplift from scaling.

Shown in Figure 27 are the training histories for BetaNet2 and BetaNet3 in isolation from benchmark models, comparing performance and training time at R=224 with R=384. Both BetaNet2 and BetaNet3 models incur additional training time cost when scaled to R=384 as evident from the horizontal shift between R=224 models and R=384 models in the bottom row of the figure. For this additional training time BetaNet3 exhibits significant improvement in accuracy, while BetaNet2 exhibits only slight improvement in accuracy.

8. DISCUSSION

Both BetaNet2 and BetaNet3 searches were successful in identifying model architectures which achieved performance competitive with or superior to the benchmark models at the baseline resolution of R=224. However, the results obtained indicate that the selected models exhibit very different scaling properties. These results are discussed as follows with regard to the main hypotheses set out in Section 3.

8.1 Hypothesis 1 - Self-similarity scaling

It has been shown that the continuous global parameterisation schemes described in Section 4.4 can be used to generate valid convolutional neural network models within the constraints of the architectural precepts adopted from EfficientNetv2. Furthermore it has been shown that these schemes, with overall model scale governed by input resolution for some otherwise fixed global parameterisation (self-similarity) produce models which increase in performance with increasing input resolution. This result echoes the findings of [Tan et al., 2020] in demonstrating viability of a global or compound scaling heuristic.

Of the two global parameterisation schemes tested, BetaNet3 demonstrates a more significant increase in model performance with increasing input resolution (Figure 27), and BetaNet2 fails to surpass the performance of the EfficientNetV2-S baseline model at input resolution R=384. This demonstrates a weakness in the approach to self-similarity scaling taken by this project.

One potential cause for this weakness is the variation in the rate at which model size scales with input resolution between the two parameterisation schemes. From Table 7 and Table 8, the BetaNet2 model scales from 8.86M parameters at R=224 to 30.1M parameters at R=384, a scaling ratio of 3.4 and nearly 1.5 times the size of the EfficientNetV2-S benchmark model. The BetaNet3 model by comparison scales from 5.32M parameters at R=224 to 15.0M parameters at R=384, a scaling ratio of 2.8.

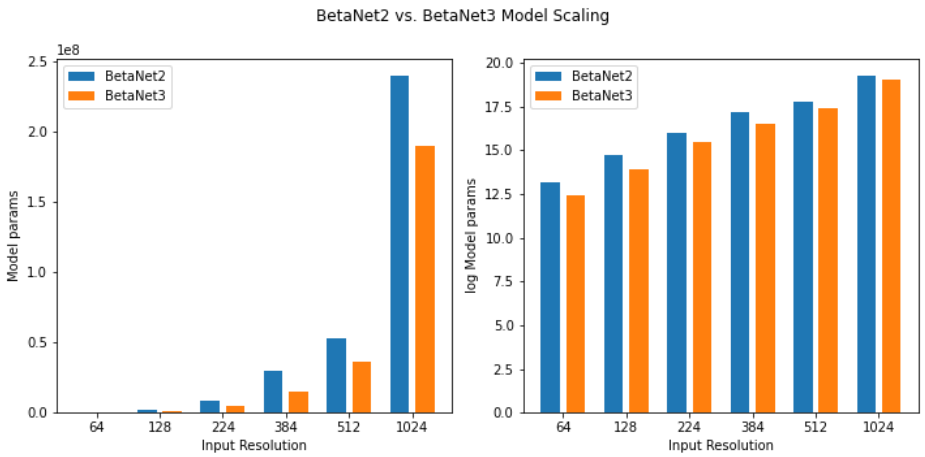


Figure 28 - BetaNet2 and BetaNet3 self-similarity scaling from R=64 to R=1,024. BetaNet2 is a much larger model at all input resolutions, and both models increase to infeasible scale at input resolution R=1,024.

The rates of scaling of BetaNet2 and BetaNet3 with increasing input resolution are explored in Figure 28. While BetaNet3 is consistently smaller than BetaNet2, both models appear to increase to an infeasible model size by R=1,024. One decision of which this is a consequence is that of requiring sufficient spatial reduction layers to reduce the spatial extent of the feature map to less than $2^{\mathbf{k}}$ pixels, where **k** is invariant to the scale of the model. This decision could be revisited by increasing **k** with input resolution.

Another potential approach to improving the scaling property of selected models is inspired by an early iteration of the search procedure developed during this project where NAS was conducted at a smaller input resolution of R=64 and then scaled up to R=224. It was quickly discovered that many models selected at R=64 produced models that were too large to train at R=224. To address this, a filtering step was introduced whereby models sampled were first tested at R=224 to confirm that they were of an acceptable parameter set and feature map size before proceeding to evaluate their performance. This step was later abandoned when searching at R=224 due to the need to allow for sufficient variation in model shape while searching. This approach could be repurposed to create a compound objective function including scaling ratio as well as model performance.

8.2 Hypothesis 2 - Bayesian Optimisation

It has been shown that Bayesian Optimisation has the potential to produce strong models given a suitably constructed search space. The key characteristic of the search

space enabling Bayesian Optimisation is for neighbouring points in the search space to correspond to similar model architectures.

While each point in the continuous parameter space corresponds to a specific model architecture, the essentially discrete nature of the model architecture means that neighboring points in the space can describe precisely the same model architecture. However, the design of the parameterisation schemes are such that neighboring architectures in the parameter space are minimally different from one another. For example, considering the parameter **Da** governing the distribution of spatial reduction layers throughout the depth of the model, varying **Da** by the minimum amount required to cause an adjustment in the model architecture while holding other parameters fixed will likely result in a single spatial reduction layer shifting up or down the model depth by a single layer.

Models selected by means of Bayesian Optimisation of a Gaussian Process (GP) model approximating the objective function have been shown to produce models which when trained to convergence demonstrate strong performance compared with benchmark models (Figure 26), as well as significantly faster training. Furthermore, the Gaussian Process Expected Improvement (GPEI) algorithm, by continually sampling points in the search space where the GP model expectation and uncertainty are such that higher performance models are statistically plausible, avoids the tendency of alternative reinforcement learning approaches for greedy exploitation of moderately successful architectural patterns. Rather, the GPEI algorithm, if permitted to continue indefinitely, could be expected to effectively map the entire search space while paying most attention to highest performing regions.

8.3 Hypothesis 3 - Selecting for training time directly

This project included training / inference time more directly in the objective function by allowing each candidate model a fixed training time budget of 5 minutes and evaluating the candidate model on the best top-1 validation set accuracy achieved in that time. This objective function / proxy task is preferable to alternatives such as training for a fixed small number of epochs and specification of a compound objective function such as that used by [Tan et al., 2020] for the following two reasons.

1. The need for deciding on a suitable structure and parameterisation of the compound objective function is avoided.
2. The practitioner can tune the search procedure to have confidence in the time that will be required to evaluate a given number of candidate models.

Models selected via NAS where the objective function includes training time exhibit significantly faster training time compared with similar or even smaller models. From Table 7 and Table 8, the selected BetaNet3 model at R=224 trains faster than the MobileNetV2 benchmark model despite having 2.3 times more parameters. Both BetaNet2 and BetaNet3 models at R=384 also train significantly faster than the EfficientNetV2-S benchmark model.

As such, it may be that inclusion of training time in the objective function as demonstrated in this project represents a stronger preference for reduced training time compared with the compound objective function approach taken by [Tan et al., 2020], and does not appear to incur penalty in terms of reduced performance as a result.

9. CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

This project has demonstrated techniques applicable to each of the three primary considerations of Neural Architecture Search (NAS) as follows.

1. **Search Space:** Designing a continuous global model parameterisation around a suitable baseline model architecture. For this project, the BetaNet2 and BetaNet3 search spaces were constructed to reflect architectural precepts of the EfficientNetV2 family of Convolutional Neural Network models. It has been demonstrated that these parameterisation schemes produce models which can be scaled up for higher accuracy, though the uplift in accuracy is sensitive to the particular architecture selected.
2. **Search Strategy:** Bayesian Optimisation of Gaussian Process models approximating the true objective function mapping model architectures to evaluation scores. It has been demonstrated that this approach reliably produces models which exhibit strong performance at a baseline scale, and is flexible enough to accommodate rejection sampling as a means to filter for models with desirable properties.
3. **Objective Function:** Integration of training time directly in the objective function by imposing a training time budget for each candidate trial. This has been shown to produce models which train much faster than comparable benchmark models, while often achieving higher accuracy. This approach also has the benefit of affording the practitioner greater control over the time required for a particular search to complete, thereby more naturally aligning with the instrumental business objectives such as planning and scheduling.

It is in a sense not surprising that NAS is able to discover architectures that outperform benchmarks with respect to a dataset that the NAS itself was based on. Yet it is generally the case that new domains will involve novel datasets and tasks which benchmark "generalist" models are not optimised for.

This project adopted CIFAR100 for the purpose of developing and integrating the techniques described above and all results presented herein are based on this dataset. CIFAR100 is a relatively small dataset, both in terms of number of examples, and image resolution, which may be a contributing factor toward the superior performance of BetaNet3 over larger benchmark models such as ResNet or

EfficinetNet. Indeed, larger benchmark models have been developed specifically for high performance on larger datasets such as ImageNet. Evaluation of the BetaNet3 model in particular on larger datasets would be required to determine whether advantages of BetaNet3 demonstrated in this work scale to larger datasets. Notwithstanding this limitation, it has been demonstrated that these techniques are effective for refinement of a strong baseline architecture for application to a specific domain, dataset, or task, where CIFAR100 may be thought of as representative of such a domain.

The selected BetaNet2 and BetaNet3 models exhibit distinct architectural features (refer Tables A1 to A4 in Appendix A), despite deliberately similar search spaces. It is speculated that this is due to the existence of multiple local maxima in the objective function landscape which a longer-running search via GPEI sampling may have exhausted. In order to have greater confidence in the objective function maxima preferred by the GP model after a given period of time, it is suggested that the initial Sobol sampling could be allowed to continue until a certain minimum density of points within the search space had been achieved, before then continuing with GPEI sampling. This would provide the GP model with more seed data and result in a better, if still coarse, approximation of the objective function from which Bayesian Optimisation of the GP model would converge more rapidly and consistently.

Two examples of continuous global parameterisation schemes have been demonstrated and evaluated in the form of BetaNet2 and BetaNet3. There are many more possible parameterisation and filtering schemes that could be tested. It could be argued that the initial challenge of Neural Architecture Search, that being that the search space is so vast, has been supplanted in this way by an analogous challenge; that of selecting a parameterisation scheme from among many possible schemes. However, each parameterisation scheme, encompassing a wide variety of models, allows the practitioner to search the architecture space much more effectively when paired with a search strategy such as Bayesian Optimisation. Ultimately, these techniques have the potential to enhance the ability of Data Science teams to discover optimal variations on strong baselines in any field.

REFERENCES

[1] Tan M, Le Q V (2020) “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. arXiv:1905.11946

[2] Tan M, Le Q V (2021) “EfficientNetV2: Smaller Models and Faster Training”. arXiv:2104.00298

[3] Howard A G, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. arXiv:1704.04861

[4] Krizhevsky A, Sutskever I, Hinton G E (2012) “ImageNet Classification with Deep Convolutional Neural Networks”. Advances in Neural Information Processing Systems 25 (NIPS 2012)

[5] Ioffe S, Szegedy C (2015) “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. arXiv:1502.03167

[6] He K, Zhang X, Ren S, Jian Sun J (2015) “Deep Residual Learning for Image Recognition”. arXiv:1512.03385

[7] Li H, Xu Z, Taylor G, Studer C, Goldstein T (2018) “Visualizing the Loss Landscape of Neural Nets”. arXiv:1712.09913

[8] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L C (2019) “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. arXiv:1801.04381

[9] Hu J, Shen L, Albanie S, Sun G, Wu E (2019) “Squeeze-and-Excitation Networks”. arXiv:1709.01507

[10] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L, Polosukhin I (2017) “Attention Is All You Need”. Advances in Neural Information Processing Systems 30 (NIPS 2017)

[11] Zoph B, Le Q V (2017) “Neural Architecture Search with Reinforcement Learning”. arXiv:1611.01578

[12] Zoph B, Vasudevan V, Shlens J, Le Q V (2018) “Learning Transferable Architectures for Scalable Image Recognition”. arXiv:1707.07012

[13] Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le Q V (2019) “MnasNet: Platform-Aware Neural Architecture Search for Mobile”. arXiv:1807.11626

[14] Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) “Algorithms for Hyper-Parameter Optimization”. Advances in Neural Information Processing Systems 24 (NIPS 2011)

[15] Bergstra J, Bengio Y (2012) “Random Search for Hyper-Parameter Optimization”. Journal of Machine Learning Research 13 (2012) 281-305

[16] Bergstra J, Yamins D, Cox D (2013) “Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures”. Proceedings of the 30th International Conference on Machine Learning, PMLR 28(1):115-123, 2013.

[17] Radosavovic I, Kosaraju RJ, Girshick R, He K, Dollar P (2020) “Designing Network Design Spaces”. arXiv:2003.13678

APPENDIX A - BETANET2 AND BETANET3 BEST ARCHITECTURES

Table A1 - Selected BetaNet2 architecture at R=224 (8.86M params)

Layer	In channels	Out channels	Expansion ratio	Stride	SE Bottleneck ratio	Repeats
CNA	3	5	3.12	2	18.14	1
FMBC	5	5	3.02	1	18.29	2
FMBC	5	25	2.93	2	18.44	1
FMBC	25	25	2.87	1	18.54	1
FMBC	25	84	2.81	2	18.64	1
MBC	84	84	2.74	1	18.74	1
MBC	84	210	2.68	2	18.84	1
MBC	210	210	2.56	1	19.03	3
MBC	210	448	2.43	2	19.23	1
MBC	448	448	2.19	1	19.63	7

Table A2 - Selected BetaNet2 architecture at R=384 (30.01M params)

Layer	In channels	Out channels	Expansion ratio	Stride	SE Bottleneck ratio	Repeats
CNA	3	5	3.12	2	18.14	1
FMBC	5	5	3.04	1	18.26	2
FMBC	5	24	2.96	2	18.38	1
FMBC	24	24	2.91	1	18.46	1
FMBC	24	80	2.86	2	18.55	1
FMBC	80	80	2.81	1	18.63	1
MBC	80	199	2.76	2	18.71	1
MBC	199	199	2.69	1	18.83	2
MBC	199	410	2.61	2	18.95	1
MBC	410	410	2.51	1	19.12	3
MBC	410	768	2.41	2	19.28	1
MBC	768	768	2.18	1	19.65	8

Table A3 - Selected BetaNet3 architecture at R=224 (5.11M params)

Layer	In channels	Out channels	Expansion ratio	Stride	SE Bottleneck ratio	Repeats
CNA	3	8	4.61	2	21.92	1
FMBC	8	30	4.60	2	21.78	1
MBC	30	59	4.60	2	21.64	1
MBC	59	92	4.59	1	21.50	1
MBC	92	129	4.59	2	21.36	1
MBC	129	129	4.58	1	21.21	1
MBC	129	211	4.58	2	21.07	1
MBC	211	211	4.55	1	20.43	8

Table A4 - Selected BetaNet3 architecture at R=384 (14.63M params)

Layer	In channels	Out channels	Expansion ratio	Stride	SE Bottleneck ratio	Repeats
CNA	3	13	4.61	2	21.92	1
FMBC	13	50	4.60	2	21.78	1
MBC	50	99	4.60	2	21.64	1
MBC	99	157	4.59	1	21.50	1
MBC	157	221	4.59	2	21.36	1
MBC	221	221	4.58	1	21.21	1

MBC	221	361	4.58	2	21.07	1
MBC	361	361	4.55	1	20.43	8

APPENDIX B - BI-WEEKLY PROGRESS REPORTS

USYD-17A

EfficientNetV2 Implementation

Adam Peaston

Project Goal

This project aims to create a library of pre-trained and/or tweaked models based on the EfficientNetV2 model architecture,

The main objectives of this project are:

1. Implement the EfficientNetV2 model in jupyter notebooks on Strong Compute's super cluster hardware in Pytorch.
2. Optimise the code for distributed training on multiple machines using Distributed Data Parallel in Pytorch, and techniques such as gradient compression.
3. Write functions to use or convert a number of well labelled large datasets (e.g. the various ImageNet datasets, Stanford Cars dataset, etc.) to be fed into the algorithm.

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

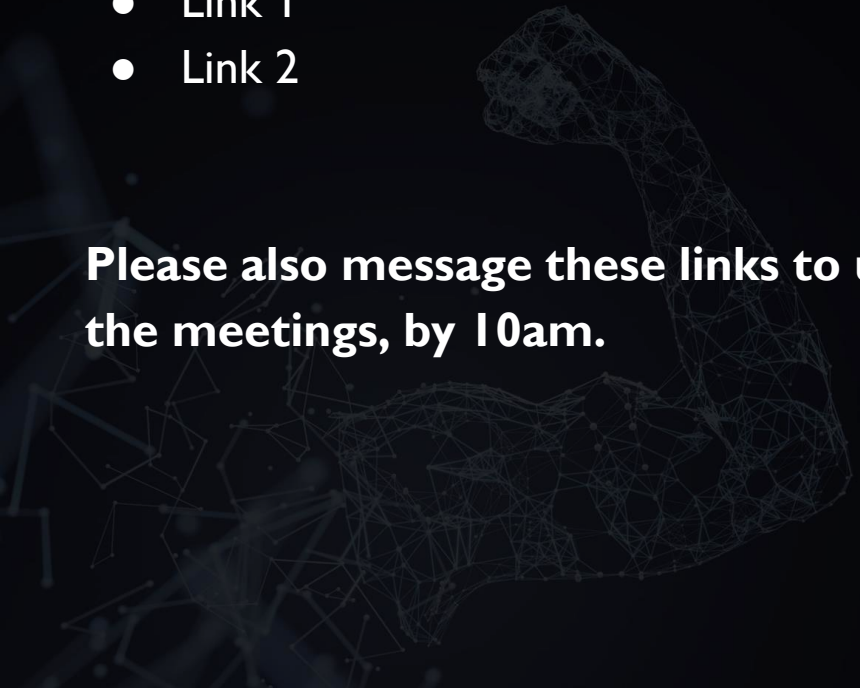
15/08/22 | WEEK 3 | MEETING 1

Video URL: https://youtu.be/9qo_R-m6ezg

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- Link 1
- Link 2



Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Complete preparatory steps from welcome email	80%	All except Google Form	Need to prepare a CV.
Read research papers	100%	Read research papers for EfficientNet and EfficientNetV2	
Set up local environment	100%	• Install Pytorch & torchvision locally • Downloaded ImageNet 2012-2017 from kaggle	Questions regarding github and remoting into SC cluster
Pytorch Tutorials	80%	• Installing Pytorch • Intro to Pytorch • MNIST • Understanding Distributed Computing (read) • Writing Distributed Applications (read) • Pytorch Docs	Generally familiar with writing models (from scratch) in Pytorch. Read DDP and did some background reading about data parallel vs. model parallel. Read through tutorials but need to read again probably a few times.
Additional Work Completed			

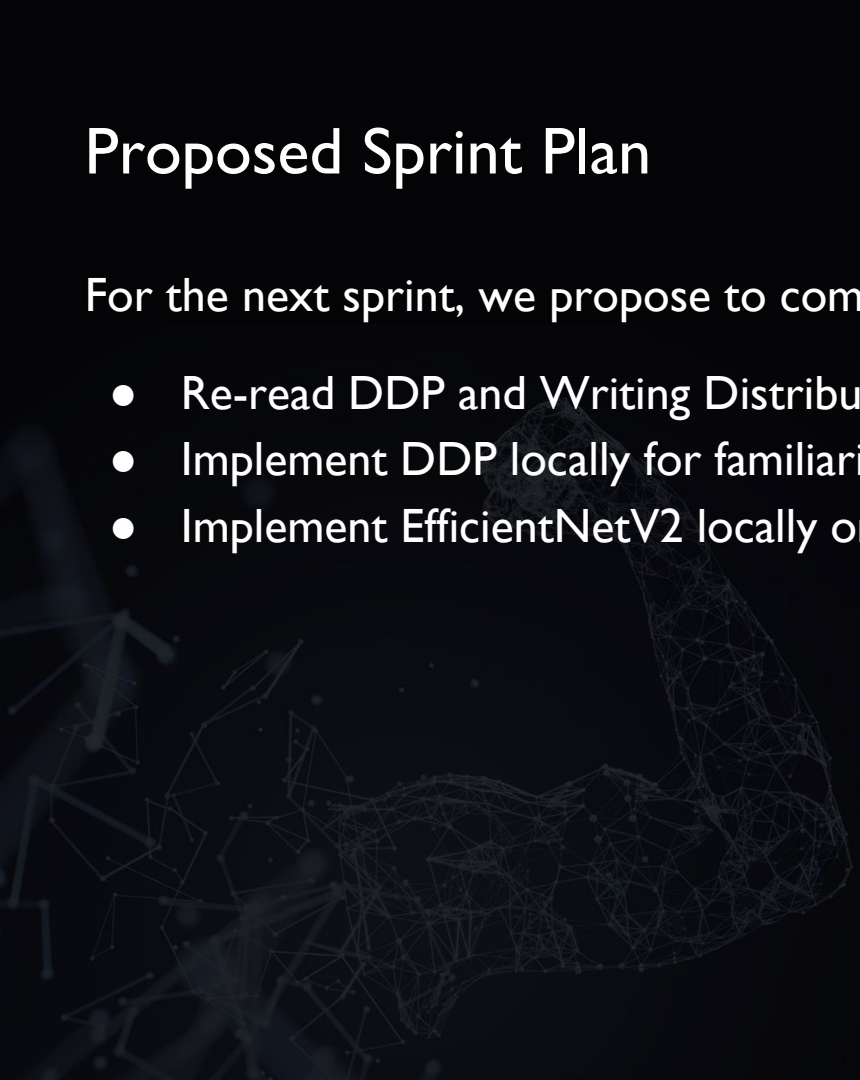
Questions

- What is SC hoping to get as a useful output of this work?
- Which datasets would SC be hoping to use?
- Is the desired output one optimised EfficientNetV2 model per large dataset?
- Usyd deliverables review:
  - Proposal Report (Week 5) - Sunday 4 September
  - Progress Report (Week 9) - Sunday 9 October
  - Presentation (Week 13) - Sunday 13 November
  - Final Report (Week 14) - Sunday 20 November
- Does SC have any guidelines for the above besides the client expectations doc (seemed more about formatting than structure?)

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Re-read DDP and Writing Distributed Applications tutorials
- Implement DDP locally for familiarity
- Implement EfficientNetV2 locally on imagenet for familiarity



Strong Compute.

Final Sprint Plan

For the next sprint, we have agreed with the client to complete the following tasks:

- Re-read DDP and Writing Distributed Applications tutorials
- Implement DDP locally for familiarity
- Implement EfficientNetV2 locally on imagenet for familiarity
- Chase down dataset of large size images ~ 1000x1000 for use

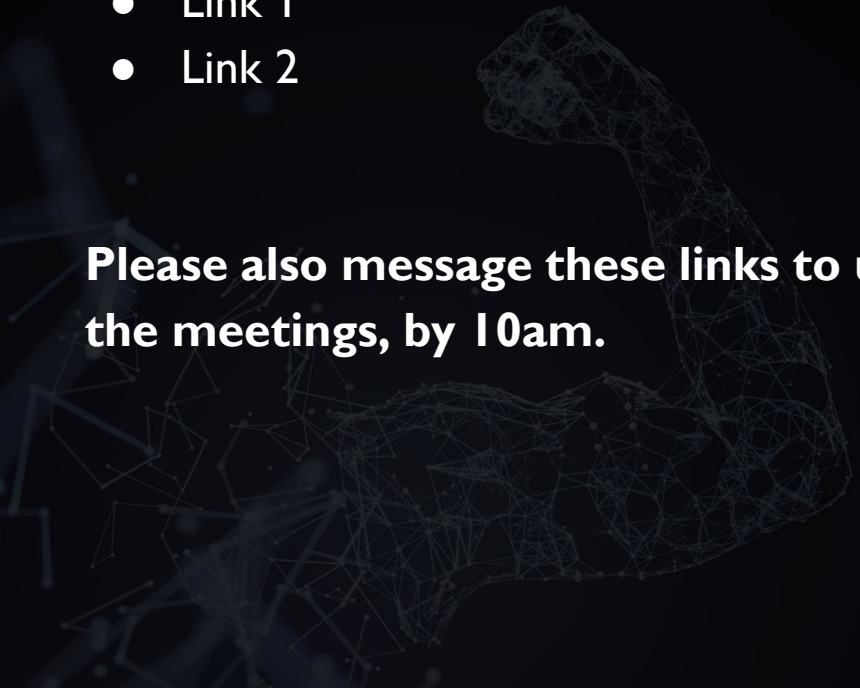

Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
15/08/22 | WEEK 3 | MEETING 2
Video URL: <>

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- Link 1
- Link 2



Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

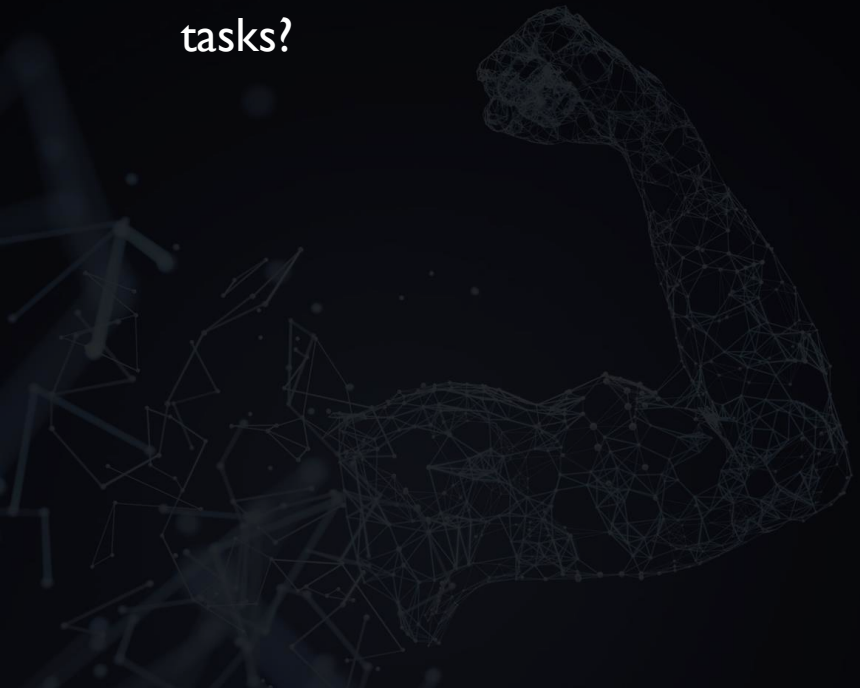
Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Re-read DDP and Writing Distributed Applications tutorials	100%	More familiar with DDP in Pytorch (tick!)	
Implement DDP locally for familiarity	20%	Transcribed code from tutorials but yet to implement on dataset	
Implement EfficientNetV2 locally on imagenet for familiarity	50%	Loaded and reviewed EfficientNetV2 (small) pre-trained model from torchvision	
Chase down dataset of large size images ~ 1000x1000 for use	100%	Installed “fiftyone” package for retrieving datasets. Downloaded 100 example images from train and validation sets of the Open Image dataset.	

Strong Compute.

Questions

- Open Image dataset more readily applicable to object detection and segmentation, need to consider application to classification or adapt EffNetV2 to one of these tasks?



Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Get EffNetV2 built from scratch
- Pass some Open Image data through that model for testing
- Implement DDP locally for familiarity

Strong Compute.

Final Sprint Plan

For the next sprint, we have agreed with the client to complete the following tasks:

- Get EffNetV2 built from scratch
- Pass some Open Image data through that model for testing
- Implement DDP locally for familiarity
- Track down and read up on Detectron
- Start to document scope for the project


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

22/08/22 | WEEK 4 | MEETING 1

Video URL: https://youtu.be/Z2pR0G77o_o

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- Draft Proposal Report:
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mItl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
• Get EffNetV2 built from scratch • Pass some Open Image data through that model for testing • Implement DDP locally for familiarity	10%	Depthwise-separable conv block prototyped. Degree of required customizability dependent on scope for NAS.	Deferred further work on code development. Focussed instead on literature review and scope definition.
Track down and read up on Detectron2	100%	Research into Detectron2 framework for possible application to project.	Requires MacOS or Linux (prohibitive). Interesting to consider, but may exclude from project scope in favour of more in-depth work on NAS.
Start to document scope for the project	100%	Draft research proposal report commenced.	Literature review focussing on efficient architectures and NAS, focussing on Bayesian Optimisation with Gaussian process or tree-parzen estimators.

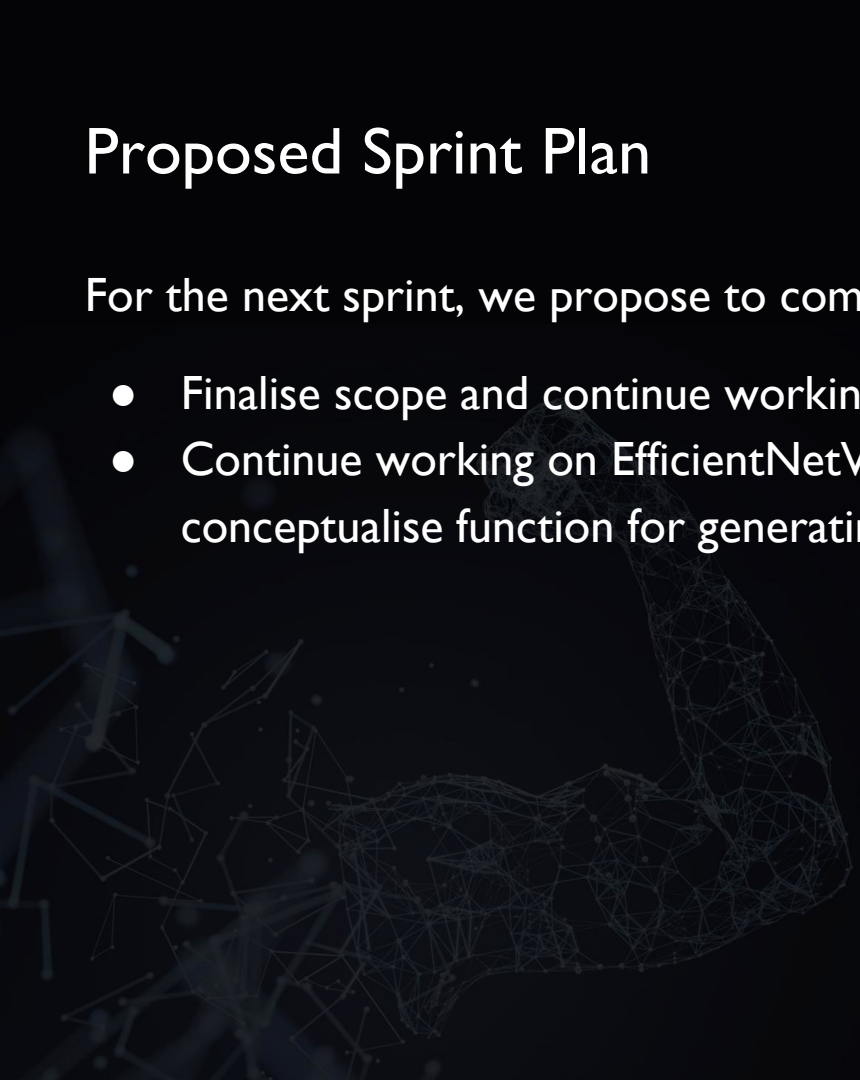
Questions

- Limiting scope to Bayesian Optimisation for NAS applied to EfficientNetV2 (or related CNN family structure), random search + Bayesian Optimisation seems to be equivalent to running AxClient() with N trials run in parallel - seems to be supported by Ax Service API.
  - Does SC have experience with Ax parallelism on SC cluster?
- Any experience compare/contrast with other packages e.g. hyperopt?

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Finalise scope and continue working on draft Proposal Report.
- Continue working on EfficientNetV2 components for hyper-parameterisation, and conceptualise function for generating model based on hyper-parameter descriptions.



Strong Compute.

Final Sprint Plan

For the next sprint, we have agreed with the client to complete the following tasks:

- Finalise scope and continue working on draft Proposal Report.
- Continue working on EfficientNetV2 components for hyper-parameterisation, and conceptualise function for generating model based on hyper-parameter descriptions.
- Spend 30mins tops and lock in a dataset


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

25/08/22 | WEEK 4 | MEETING 2

Video URL: https://youtu.be/_qoErcykAgQ

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- Draft Proposal Report:
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mItl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

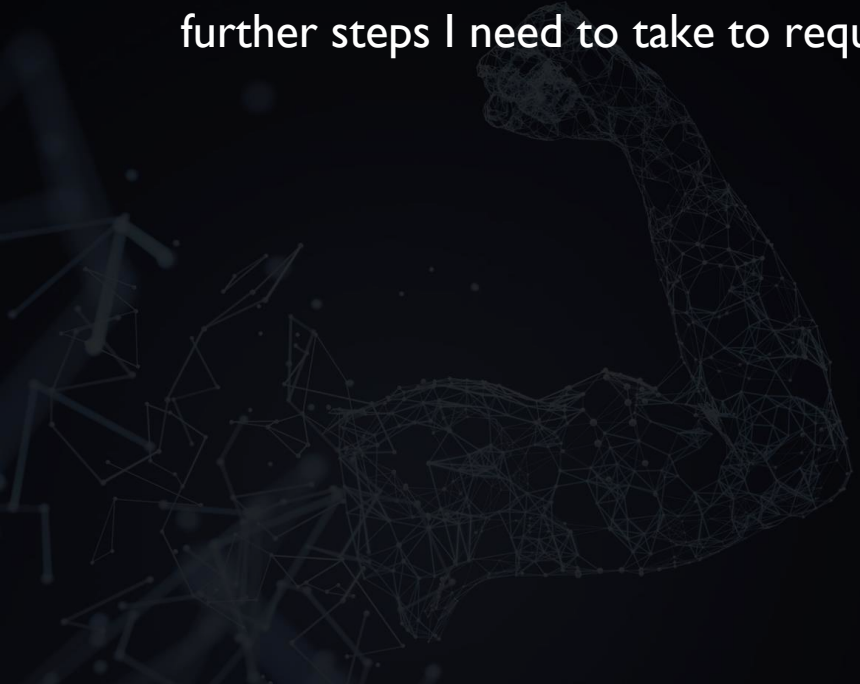
Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Finalise scope and continue working on draft Proposal Report.	50%	Scope coming together. Locking in focus on NAS with SMBO using either Ax (Gaussian Process) or Hyperopt (TPE).	Yet to land on/finalise structure of search space. EffNet searched the same space as MnasNet, “Factorized Hierarchical”. May instead use TPE with bias encoding EffNet.
Continue working on EfficientNetV2 components for hyper-parameterisation, and conceptualise function for generating model based on hyper-parameter descriptions.	100%	MBConv and Fused-MBConv layers prototyped and able to be assembled into end-to-end model architectures.	
Spend 30mins tops and lock in a dataset	100%	ImageNet locked in. Full dataset downloaded and functions prepared for ETL.	

Questions

- ImageNet is big, and will need establishment on SC infrastructure.
- Note from Cian last Friday indicated delay in provisioning SC access. Are there further steps I need to take to request access?



Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Finalise scope and ***finish*** draft Proposal Report.
- Start work on developing functions for auto-generating a network based on search space parameterisation.

Strong Compute.

Final Sprint Plan

For the next sprint, we have agreed with the client to complete the following tasks:

- Finalise scope and ***finish*** draft Proposal Report.
- Start work on developing functions for auto-generating a network based on search space parameterisation.

Strong Compute.


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
29/08/22 | WEEK 5 | MEETING 1
Video URL: https://youtu.be/Wqp1VKDj4pk

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- Draft Proposal Report:
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mItl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

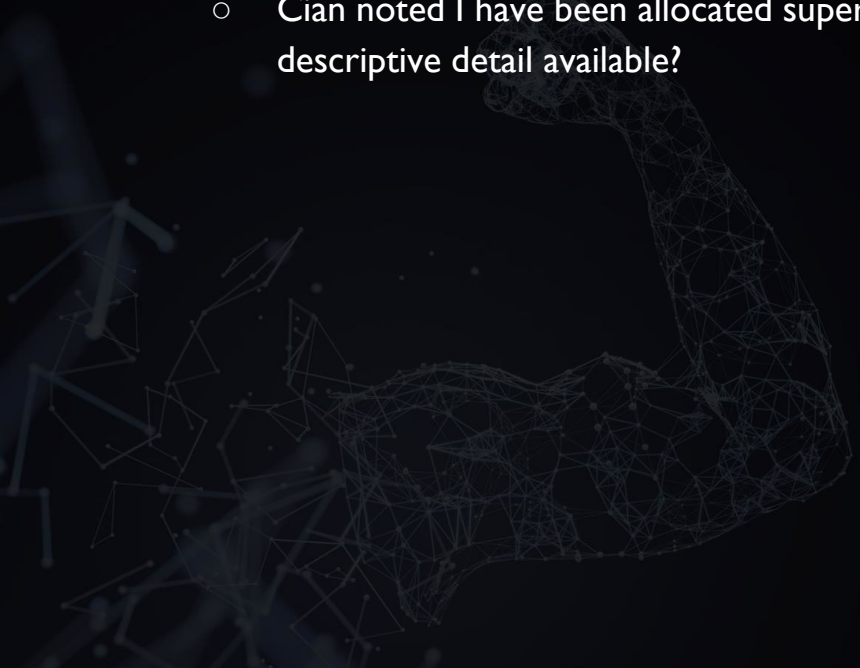
Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Finalise scope and finish draft Proposal Report.	90%	Literature review content written, and previously spent time refining scope in terms of search space, search strategy, and objective function.	Need to spend this week documenting and polishing remaining sections (resources, expected outcomes, schedules).
Start work on developing functions for auto-generating a network based on search space parameterisation.	30%	Started writing the model class which will generate the model architecture based on numeric parameters passed.	Work left to do in developing this model class and general implementation, but will have time to focus on that after proposal report is completed.

Questions

- Proposal report calls for documentation of resources (i.e. hardware and other materials), is there any boilerplate description available for the SC super cluster?
    - Cian noted I have been allocated supercluster sc8 with access to sc9 and sc10, is there more descriptive detail available?



Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Finish Proposal Report ready for submission on Sunday 4 September.
- Access provisioned SC supercluster and familiarise with connecting, setting up, and instantiating required datasets.
- Continue working on model class and implementation.

Final Sprint Plan

For the next sprint, we have agreed with the client to complete the following tasks:

- Finish Proposal Report ready for submission on Sunday 4 September.
- Access provisioned SC supercluster and familiarise with connecting, setting up, and instantiating required datasets.
- Continue working on model class and implementation.


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

01/09/22 | WEEK 5 | MEETING 2

Video URL: https://youtu.be/yXfwg7UXDCE

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- Draft Proposal Report:
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mItl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Finish Proposal Report ready for submission on Sunday 4 September.	100%	Completed Proposal Report!	Might do one or two further read-through and polish, but essentially done.
Access provisioned SC supercluster and familiarise with connecting, setting up, and instantiating required datasets.	75%	Accessed SC8 via windows command prompt, able to manipulate directories and files inside “usyd-17a” subdirectory.	Working through setting up access via VSCode to be able to work in Jupyter Notebook environment.
Continue working on model class and implementation.	30%	Incremental progress, though de-prioritised this week in favour of the above tasks.	Invested most of my time in finishing proposal report to make sure that’s done and out of the way. Remainder of this week to be spent on env set -up and code development.

Questions

- Developing model class to generate model according to procedure outlined in proposal report; for MVP would it be sufficient to record a walk through of this procedure, that class structure, and demonstrate model instantiation and inference on a batch of data?
- Is there a delivery deadline for the MVP? Is there a submission protocol, or would it be OK to post my video in Discord?

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Finish environment set up and access to Jupyter Notebooks on SC8.
- Complete model class prototype and record MVP video.

Strong Compute.

Final Sprint Plan

For the next sprint, we have agreed with the client to complete the following tasks:

- Finish environment set up and access to Jupyter Notebooks on SC8.
- Complete model class prototype and record MVP video.

Strong Compute.


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

05/09/22 | WEEK 6 | MEETING 1

Video URL: https://youtu.be/lo2rO6emaYo

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Finish environment set up and access to Jupyter Notebooks on C8.	100%	Set up with access to C8, Jupyter Lab running nicely, GPUs accessible.	Only remaining blocker seems to be inability to properly install fiftyone in container. Have not yet resolved dependency issues.
Complete model class prototype and record MVP video.	100%	Model class prototype operational. NAS training loop functional. Recording MVP video before this meeting.	Need to consider how to accelerate with parallelism immediately.
Finalise and submit Proposal Report.	100%	Proposal Report in 👍	Next up: Progress Report… due Sunday October 9.

Strong Compute.

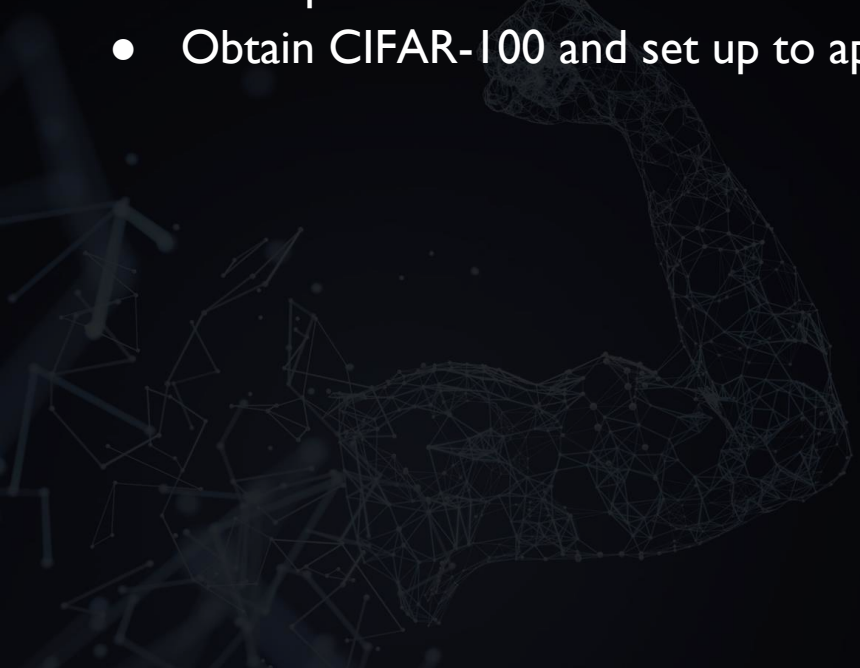
Questions

- As a potential action to train candidate models faster, considering amending the plan to use CIFAR-100 instead of ImageNet for NAS proxy task, then test on ImageNet for selected architectures. As a result, attempted to install fiftyone and obtain dataset that way. Ran into issues loading fiftyone, seemingly dependency issues with “cv2” or “opencv-python” package (same package). Can I ask for guidance on this?
- Would like also to start developing implementation of data parallelism (also to help reduce training time). If there is anything that can be shared to give me a head start setting this up on C8 that would be awesome.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Attempt a multi-trial NAS for meaningful number of epochs and obtain results.
- Obtain CIFAR-100 and set up to apply to NAS proxy-task.

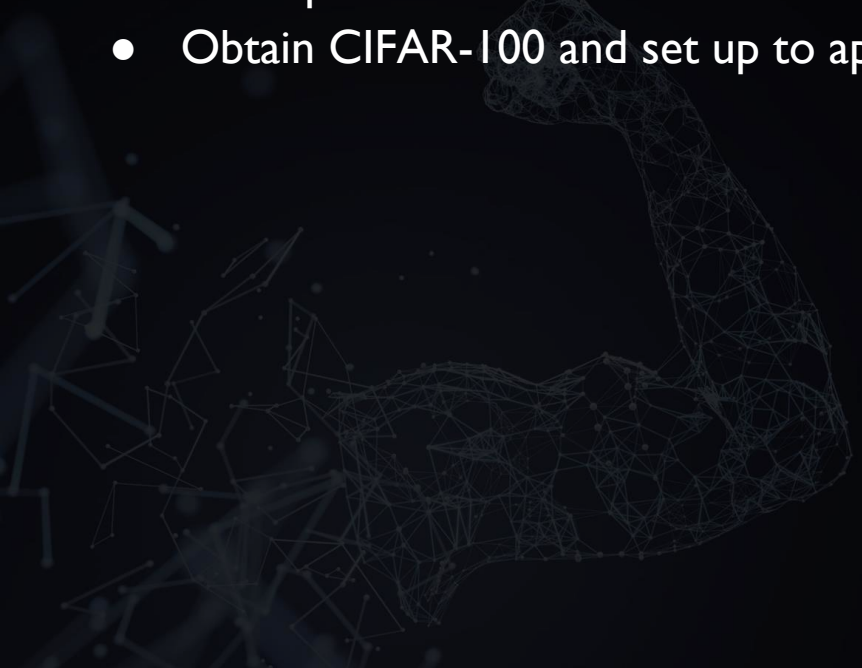


Strong Compute.

Final Sprint Plan

For the next sprint, we have agreed with the client to complete the following tasks:

- Attempt a multi-trial NAS for meaningful number of epochs and obtain results.
- Obtain CIFAR-100 and set up to apply to NAS proxy-task.



Strong Compute.


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
07/09/22 | WEEK 6 | MEETING 2
Video URL: https://youtu.be/dgJUG5ATMxs

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Attempt a multi-trial NAS for meaningful number of epochs and obtain results..	100%	NAS with 8 numeric range variables successfully run over 100 trials on CIFAR100 dataset.	Each trial run for 20 epochs only, not to convergence. Top1 accuracy not terribly impressive.
Obtain CIFAR-100 and set up to apply to NAS proxy-task.	100%	As above.	

Results of first NAS trial

Figure right demonstrates that the NAS procedure is working to progressively propose better parameters. Something curious happens in later trials where the loss spikes for some models, reason unknown. Maybe oscillating near a minimum in the loss landscape with steep sides.

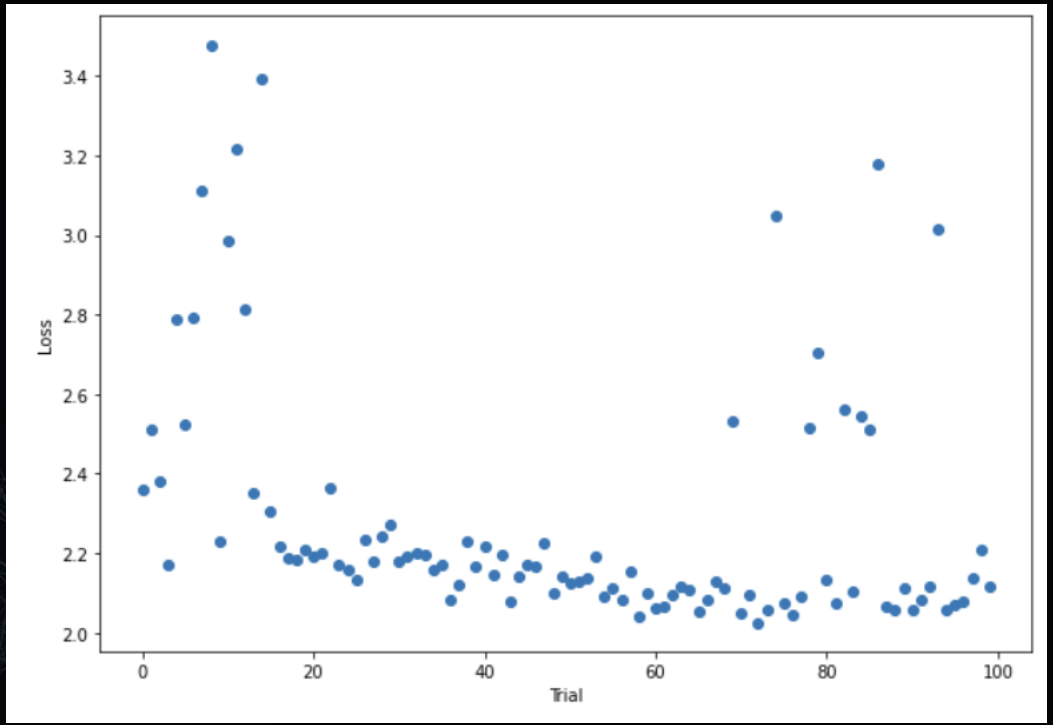
Referring to best parameters below, the search returns recommendation B=1.0, suggesting that Fused-MBConv layers are preferable throughout the network.

Best parameters recommended by this trial:

{'K': 2.245, 'Da': -0.511, 'Db': -0.740, 'G': 3.740, 'B': 1.0, 'E': 4.870, 'S': 12.650, 'LR': 0.003}

Best top-1 accuracy: 47.57%

Best top-5 accuracy: 77.56%



Strong Compute.

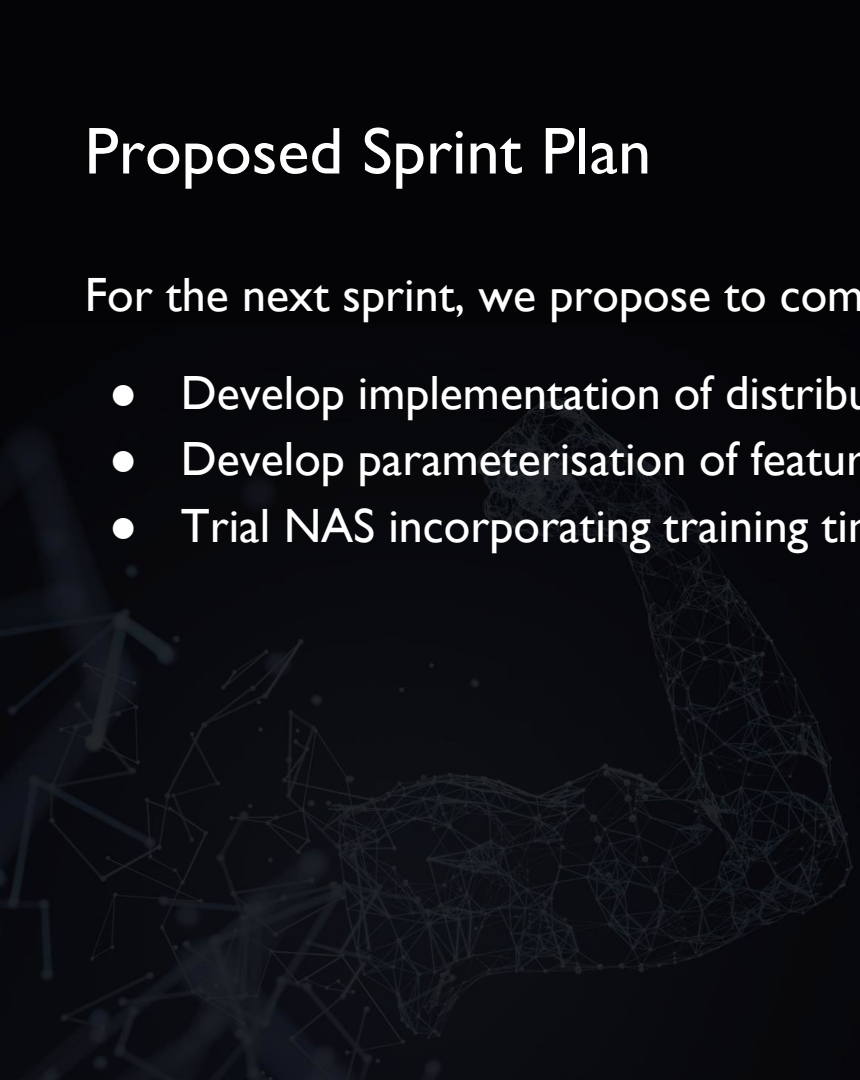
Questions

- Made a change to the approach taken to parameterising the search space; discarded dirichlet distribution in favour of beta distribution, reducing the parameters to search from N to 2 at the expense of some degree of expressivity (feels like a reasonable trade).
- Initial model generating procedure / NAS search space prescribed model layers with channels determined by logarithmic interpolation from input feature map size to latent feature map size. Considering application of beta distribution to parameterise a search space over the shape of this interpolation instead.
- Do you know any rules of thumb or heuristics, with or without sources, to suggest a sensible constraint on this aspect of the search space?

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Develop implementation of distributed training.
- Develop parameterisation of featuremap interpolation applicable to NAS.
- Trial NAS incorporating training time into objective function.

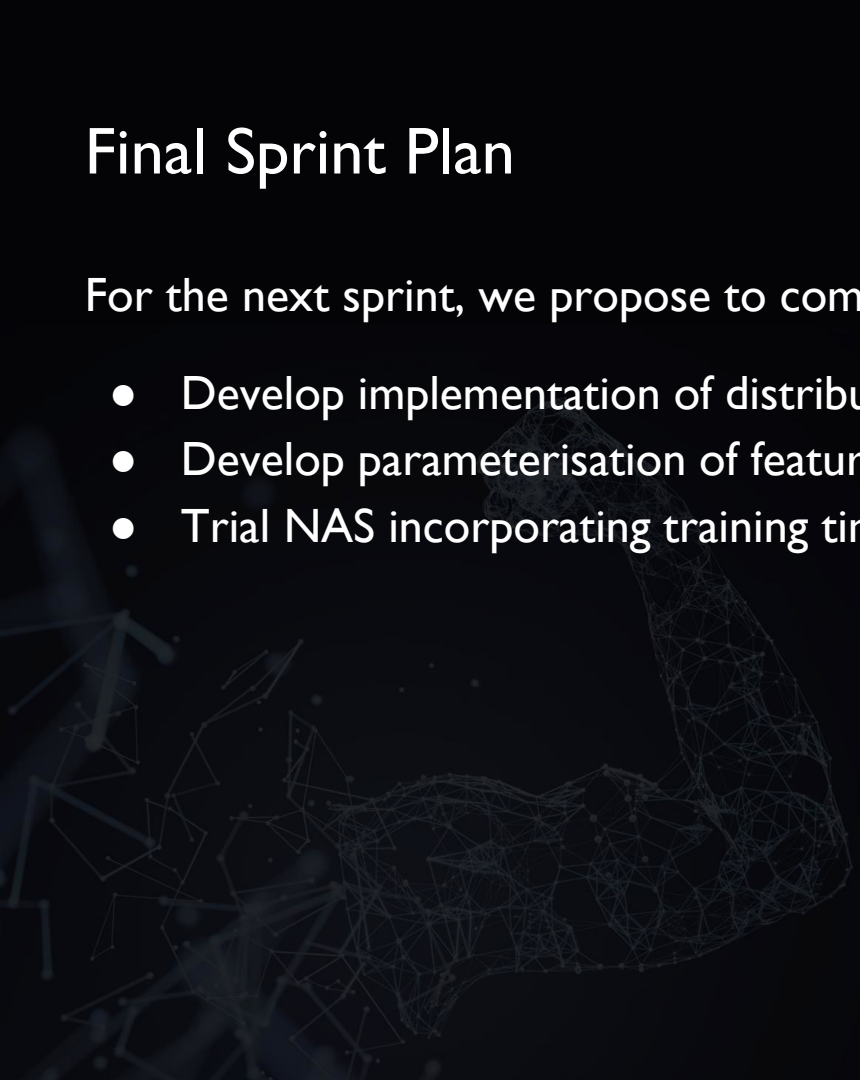


Strong Compute.

Final Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Develop implementation of distributed training.
- Develop parameterisation of featuremap interpolation applicable to NAS.
- Trial NAS incorporating training time into objective function.



Strong Compute.


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
12/09/22 | WEEK 7 | MEETING 1
Video URL: https://youtu.be/9N80G2aC-Ck

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Develop implementation of distributed training.	50%	Replicated example DDP implementation on EfficientNet model applied to CIFAR100.	Further work needed adapt training / evaluation functions and Ax implementation for DDP.
Develop parameterisation of featuremap interpolation applicable to NAS.	100%	Block depth and width now parameterised with beta distributions for a total of 4 naturally bounded numeric parameters to search.	Also upgraded reduction-step allocation algorithm for better effectiveness.
Trial NAS incorporating training time into objective function.	75%	Training loop is logging average epoch training time.	With new parameterization (above) and yet to finalize implementation of DDP, need to gather more data on unconstrained training times.

Questions

- Looks like DDP is not designed to operate in a notebook environment (I found commentary online to this effect as well when investigating DDP previously). Is there any way you know of to easily implement DDP in notebook?
- Alternative is to write .py script to build, train, and evaluate model based on passed parameters generated in notebook.
- I'm new to multiprocessing in general, any guidance or examples you could share on how to retrieve results from Processes would be appreciated.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Integrate implementation of distributed training.
- Gather data on unconstrained per-epoch training times with DDP.
- Trial NAS incorporating training time into objective function.
- Create MobileNet & ResNet50 comparators.


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

15/09/22 | WEEK 7 | MEETING 2

Video URL: https://youtu.be/dRofrT7snD8

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Integrate implementation of distributed training.	100%	Finished integrating DDP into training pipeline.	Running nicely as a throw-to from notebook.
Create MobileNet & ResNet50 comparators.	50%	Developed ResNet34 comparator training script.	Applying to CIFAR-100, instantiated ResNet34 for more direct application of 34x34 images.
Gather data on unconstrained per-epoch training times with DDP & trial NAS incorporating training time into objective function.	0%	Before proceeding with full NAS run, had hoped to see higher performance from models.	EfficientNet & ResNet models both achieving ~ 50% max top1 accuracy. Time spent instead seeking improvement before proceeding.

Questions

- One question; I'm finding that for baseline models (efficientnet_b0 for example), the following training pattern occurs*
  - 1. Training and validation loss reduces, validation accuracy increases
  - 2. Training loss reduces, validation loss increases, validation accuracy increases
- I'm wondering if it is sensible to apply early stopping based on validation/test accuracy rather than loss?

* I haven't yet split the training set into train/test so I'm using validation loss/accuracy as potential early stopping criteria (I can fix this in future by splitting train into train/test instead)

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Develop ResNet50 (instead of ResNet34) and MobileNet implementations, use image resizing for applicability to CIFAR-100.
- Work on means to apply time budget approach to objective function (great idea), need a means for message passing between devices that training time has been reached.
- Run a full NAS on ViolinNet and generate parallel axis plots.

Final Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Develop ResNet50 (instead of ResNet34) and MobileNet implementations, use image resizing for applicability to CIFAR-100.
- Work on means to apply time budget approach to objective function (great idea), need a means for message passing between devices that training time has been reached.
- Run a full NAS on ViolinNet and generate parallel axis plots.


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
19/09/22 | WEEK 8 | MEETING 1
Video URL: https://youtu.be/vGqHzeqcgGE

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Develop ResNet50 (instead of ResNet34) and MobileNet implementations, use image resizing for applicability to CIFAR-100.	100%	Now have ResNet50, MobileNetV2, EfficientNet, and EfficienNetV2 baselines complete.	Huge improvement in performance from strong augmentation and learning rate scheduling.
Work on means to apply time budget approach to objective function (great idea), need a means for message passing between devices that training time has been reached.	100%	Implemented per-trial time budget of 5 mins to complete 100 trials in ~8.5 hours. Training terminated at end of epoch when budget is exhausted.	Budget only counts toward training steps, not evaluation steps, seems fair. Not seeing huge variance in number of epochs training for very different size models?
Run a full NAS on ViolinNet and generate parallel axis plots.	100%	Ran 80 trials (out of 100 scheduled) before SC8 failed again last night. Able to recover results, presented next slide.	Repeated failures of SC8 becoming an issue for progress.

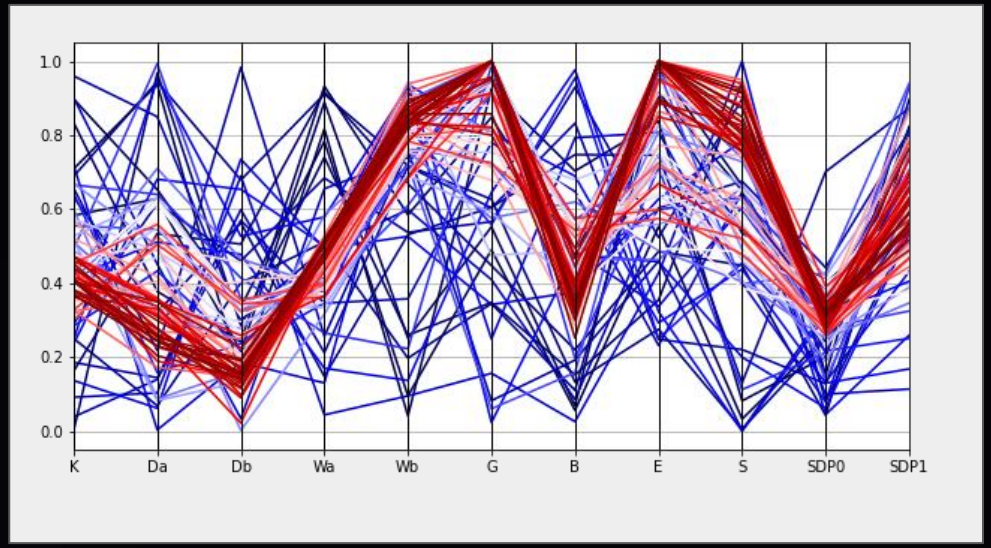
Results of NAS experiment

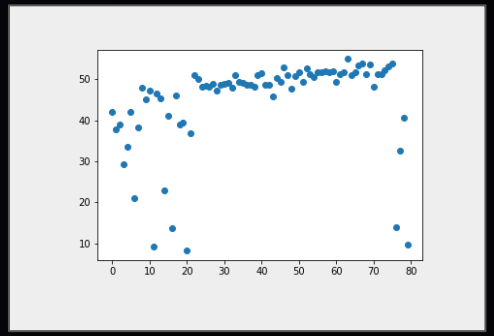
This experiment is a first attempt at the main procedure for this work; run trials on small-scale proxy task to discover good self-similarity kernel, then scale up for higher performance. The proxy task is training on R=32 CIFAR100 images with no up-scaling for a training time budget of 5 mins.

Figure top right shows parallel axis plot with one spanning line per trial. Colour indicates relative trial performance measured by top-1 accuracy, with blue indicating below-average for the experiment and red indicating above average. The max top-1 accuracy was 55%.

These results indicate the need to explore higher ranges for G (latent vector size multiple of R) and E (Fused-MBConv / MBConv expansion ratio). All other parameters appear to be converging.

However! The hyper-parameters recommended by this experiment produce an R=224 model size of over 1B parameters, clearly infeasible. Easy to scale back by reducing depth and latent vector size, but would be good to think about constraining the search space to hyper-parameters that give rise to sensibly-sized large scale models.





Strong Compute.

Questions

- SC8 seems to have ongoing issues, is there anything I can do from my end to avoid cluster failures occurring again?

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Work on AI day presentation
- Run another NAS experiment exploring higher G and E ranges, hopefully run for 100+ trials.
- Train scaled-up model (R=224) based on best kernel found in above second experiment.
- Run another NAS experiment setting R=64 to compare best self-similarity kernel between R=32 and R=64.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Work on AI day presentation
- Run another NAS experiment exploring higher G and E ranges, hopefully run for 100+ trials.
- Train scaled-up model (R=224) based on best kernel found in above second experiment.
- Run another NAS experiment setting R=64 to compare best self-similarity kernel between R=32 and R=64.


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

22/09/22 | WEEK 8 | MEETING 2

Video URL: https://youtu.be/JmtvR3GY5vo

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

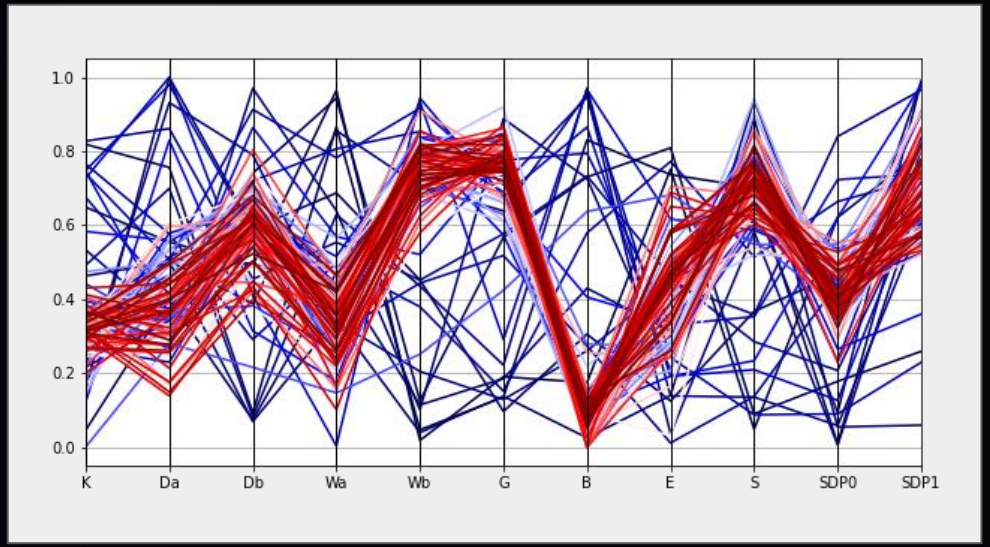
Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Work on AI day presentation.	100%	Draft slides with Ben for review.	Pumped for gokarting!
Run another NAS experiment exploring higher G and E ranges, hopefully run for 100+ trials.	75%	NAS run exploring higher G and E ranges.	Cluster crashed again preventing reach of 100+ trials.
Train scaled-up model (R=224) based on best kernel found in above second experiment.	0%	All experimental effort has been spent trying to get a 100+ trial NAS without success yet.	Failures of SC8 and SC1 suggestive that my code is responsible somehow.
Run another NAS experiment setting R=64 to compare best self-similarity kernel between R=32 and R=64	0%	As above.	

Results of NAS experiment

Parallel axis plot shows G and E parameters converging on optimal value within ranges searched with NAS (~80 trials).

Next step is simply to work out how to avoid crashing the SC clusters.



Strong Compute.

Questions

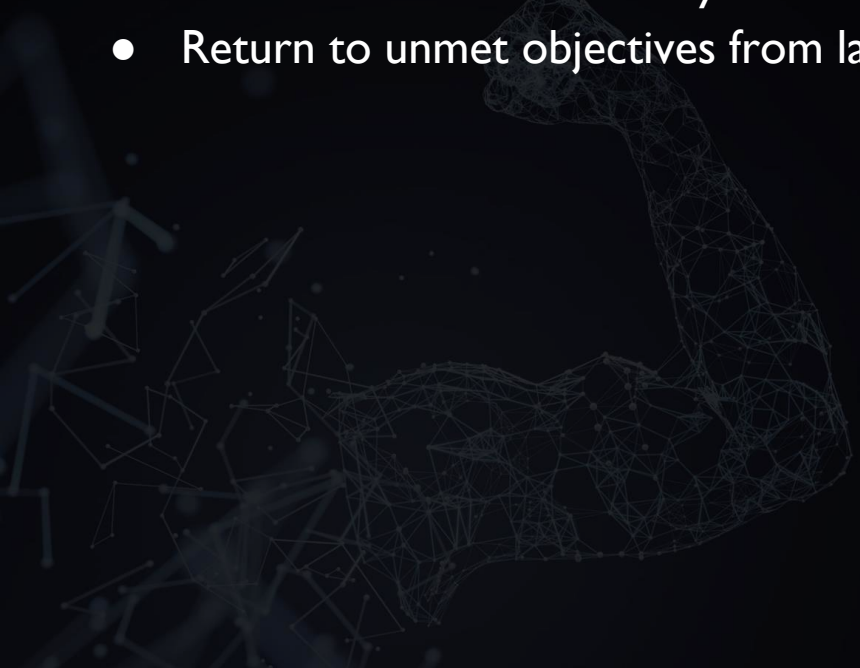
- No questions this week!

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Run NAS with 2 GPUs only to mitigate likelihood of crash.
- Return to unmet objectives from last sprint based on results of that NAS.

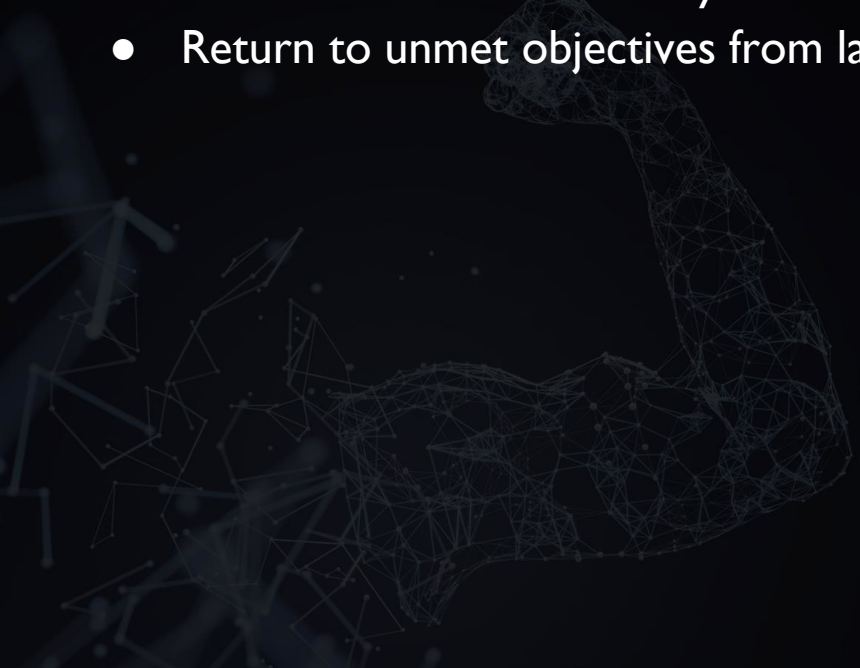


Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Run NAS with 2 GPUs only to mitigate likelihood of crash.
- Return to unmet objectives from last sprint based on results of that NAS.



Strong Compute.


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
26/09/22 | WEEK 9 | MEETING 1
Video URL: https://youtu.be/7Wxu7FWwD3E

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Run another NAS experiment exploring higher G and E ranges, hopefully run for 100+ trials.	100%	Experiment run and data gathered, demo next slide.	Multiple performant modalities emergent.
Train scaled-up model (R=224) based on best kernel found in above second experiment.	75%	Scaling procedure works as expected to produce self-similar model for arbitrary input dimension.	Small models selected by NAS scale to astronomical number of params.
Run another NAS experiment setting R=64 to compare best self-similarity kernel between R=32 and R=64.	0%	Instead pursued adjustment to NAS based on notes above.	Expecting to run R=64 NAS for comparison after sensible-scaling NAS is successful.

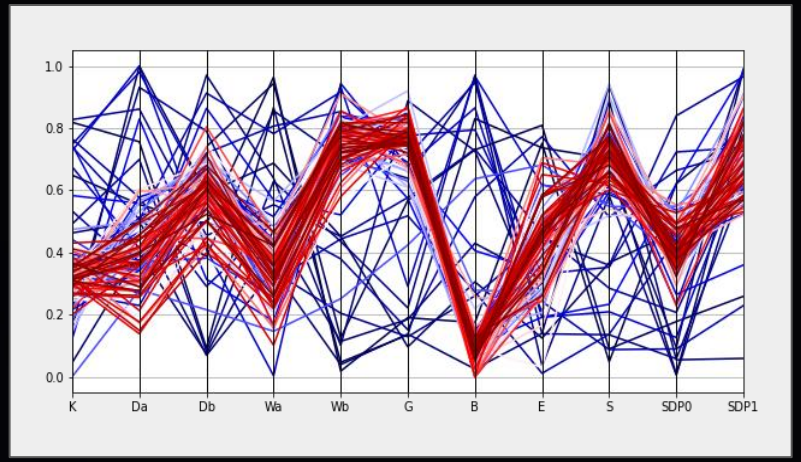
Results of NAS experiment

Running for more trials has the potential to discover multiple performant configurations. Some of these configs may scale better than others or have other properties by which they could be prioritised.

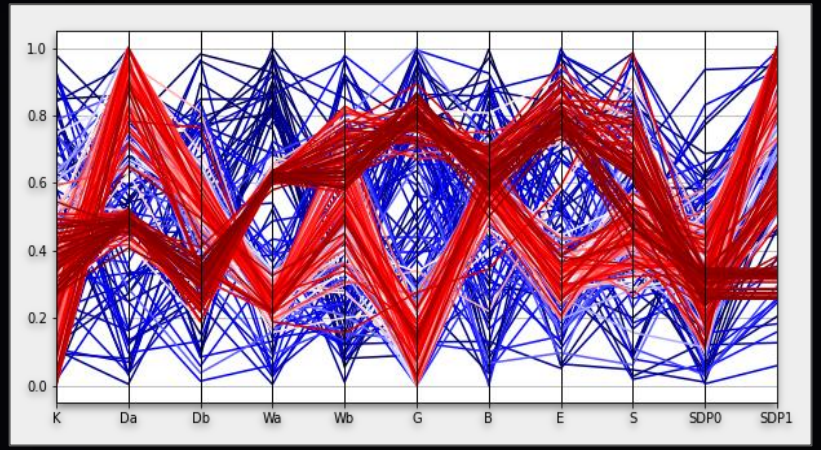
Experiment 2 preferred configs don't resemble Experiment 1 preferred configs really at all. Problem?

Preferred model config from Experiment 2 at R=32 scales to +1B params at R=224, not feasible. Need to include look-ahead filter to evaluate scaling properties of config as part of NAS.

Experiment 1 (~80 trials)



Experiment 2 (~250 trials)



Questions

- No questions this week!

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Adjust search space to focus on param ranges which independently place R=224 model size in feasible range.
- Include filter stage in NAS to reject proposed model configs producing R=224 models with > 50M params.
- Run NAS at R=32 and R=64 to compare results. Seems likely that different preferred configs will be found unless NAS is run for +250 trials each.

Final Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Adjust search space to focus on param ranges which independently place R=224 model size in feasible range.
- Include filter stage in NAS to reject proposed model configs producing R=224 models with > 50M params.
- Run NAS at R=32 and R=64 to compare results. Seems likely that different preferred configs will be found unless NAS is run for +250 trials each.


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
29/09/22 | WEEK 9 | MEETING 2
Video URL: https://youtu.be/vCtvyBMLXk8

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

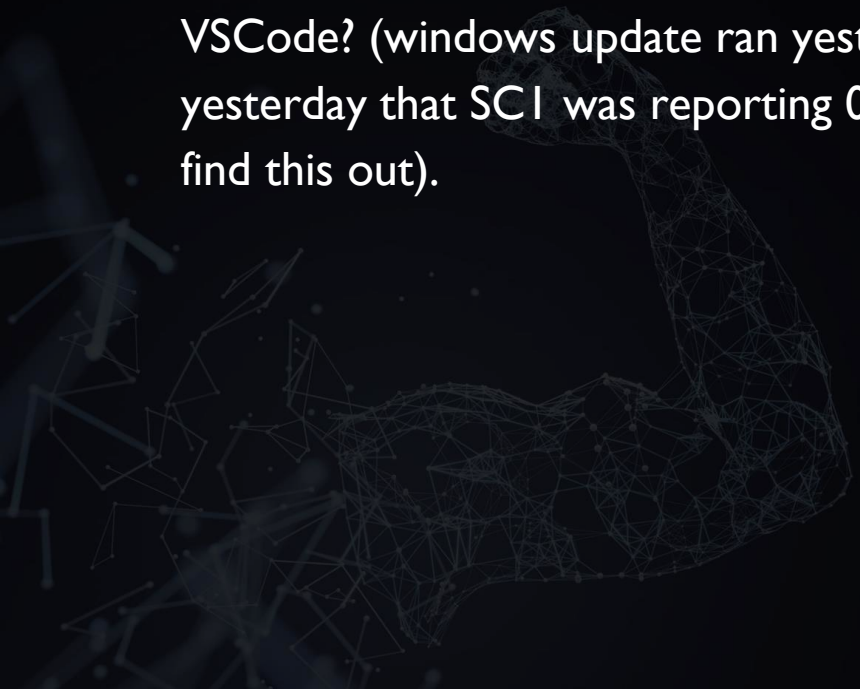
Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Adjust search space to focus on param ranges which independently place R=224 model size in feasible range.	100%	Param ranges more regularly produce feasible R=224 scale models.	Experiment not yet complete.
Include filter stage in NAS to reject proposed model configs producing R=224 models with > 50M params.	75%	NAS logging completion of trial with target = 0 (worst performance) if R=224 model is larger than 50M params.	Experiment not yet complete.
Run NAS at R=32 and R=64 to compare results. Seems likely that different preferred configs will be found unless NAS is run for +250 trials each	0%	Not yet completed experiment at R=32.	Experiment not yet complete.

Questions

- Issues encountered with SSH to SC1. Wireguard running, VSCode opened, attempt to SSH to SC1 and no password prompt appears. Could this be an issue with VSCode? (windows update ran yesterday) Or an issue with SC1? (noted in Discord yesterday that SC1 was reporting 0 GPUs available, though was able to SSH in to find this out).



Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Run NAS at R=32 and R=64 to compare results. Seems likely that different preferred configs will be found unless NAS is run for +250 trials each.
- Scale up to R=224 and compare results with ResNet50, MobileNetV2, EfficientNet and EfficientNetV2 benchmarks.


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

3/10/22 | WEEK 10 | MEETING 1

Video URL: https://youtu.be/dOvY7c0TtbI

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

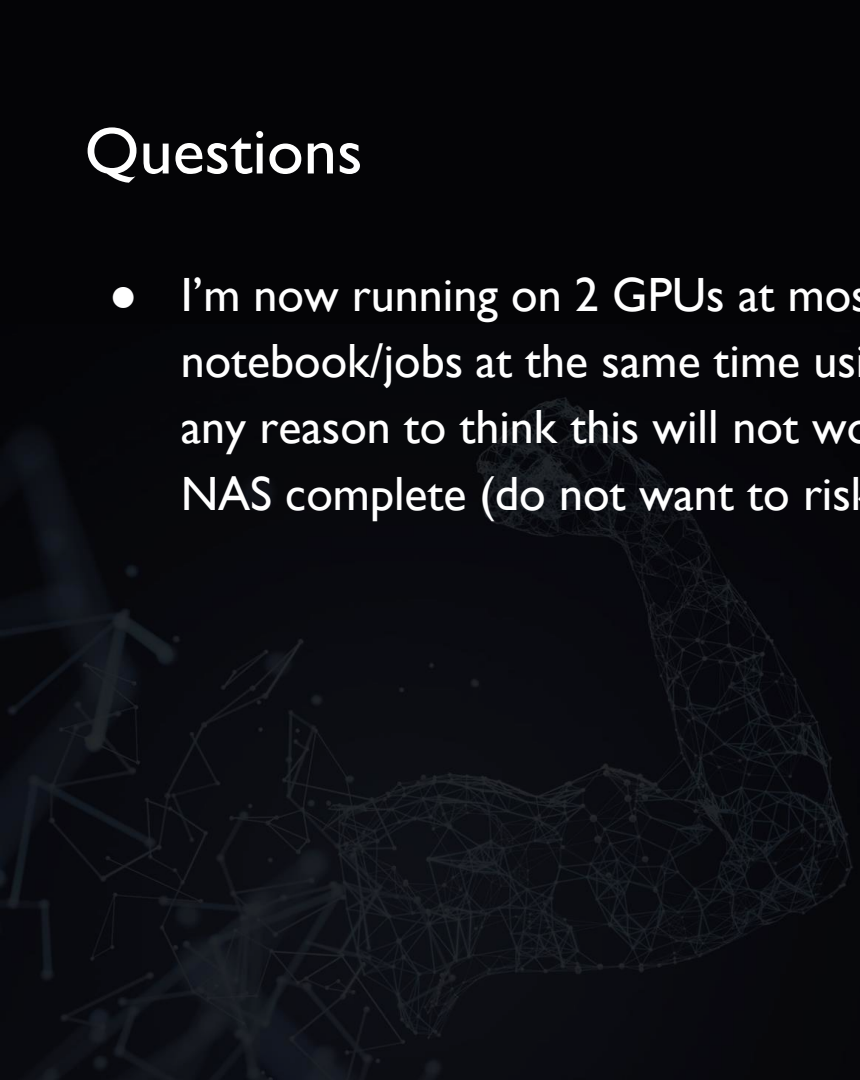
Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Run NAS at R=32 and R=64 to compare results. Seems likely that different preferred configs will be found unless NAS is run for +250 trials each.	50%	R=32 experiment near completion (~75% complete running now).	Some delay due to VSCode issue, then some NAS procedure correction.
Scale up to R=224 and compare results with ResNet50, MobileNetV2, EfficientNet and EfficientNetV2 benchmarks.	75%	Ran R=32 experiment and scaled up preferred model. R=224 model had acceptable params but encountered GPU memory issues when training. Second pre-filter included in NAS to reject models unable to actually train at R=224.	Updated R=32 NAS nearing completion now. Lots of training for benchmarks etc. still required.

Questions

- I'm now running on 2 GPUs at most for a given notebook/job. If I run two more notebook/jobs at the same time using 2 GPUs each (total 6 GPUs available) is there any reason to think this will not work? I can run this experiment as soon as R=32 NAS complete (do not want to risk crashing this job).



Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Complete R=32 NAS (nearly finished now).
- Scale up to R=224 and train to convergence.
- Re-train benchmark models (2 GPUs constraint requires re-training for like-for-like comparison).
- Run R=64 NAS and compare with R=32 NAS results (potentially scale up R=64 kernel to R=224 and compare with R=32 kernel scaled model).
- Finish and submit progress report by October 9th.


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

6/10/22 | WEEK 10 | MEETING 2

Video URL: https://youtu.be/KtC2Wbu3J_E

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Progress Report (Due Sunday October 9):
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Complete R=32 NAS (nearly finished now), scale up to R=224 and train to convergence.	100%	Completed experiment, scaled up to R=224 trained to convergence.	Not wildly impressive performance ~ 66% top1 accuracy.
Re-train benchmark models (2 GPUs constraint requires re-training for like-for-like comparison).	100%	Completed training of all 4 benchmark models.	
Run R=64 NAS and compare with R=32 NAS results (potentially scale up R=64 kernel to R=224 and compare with R=32 kernel scaled model).	75%(?)	Running R=64 NAS with an additional tweak shooting for R=224 performance gain last night.	This morning SC1 is unreachable by SSH. Not good.
Finish and submit progress report by October 9th.	95%	Report document mostly drafted. Need to include one figure and references.	Due this Sunday 9th, if you are able to give any feedback before then appreciated.

Questions

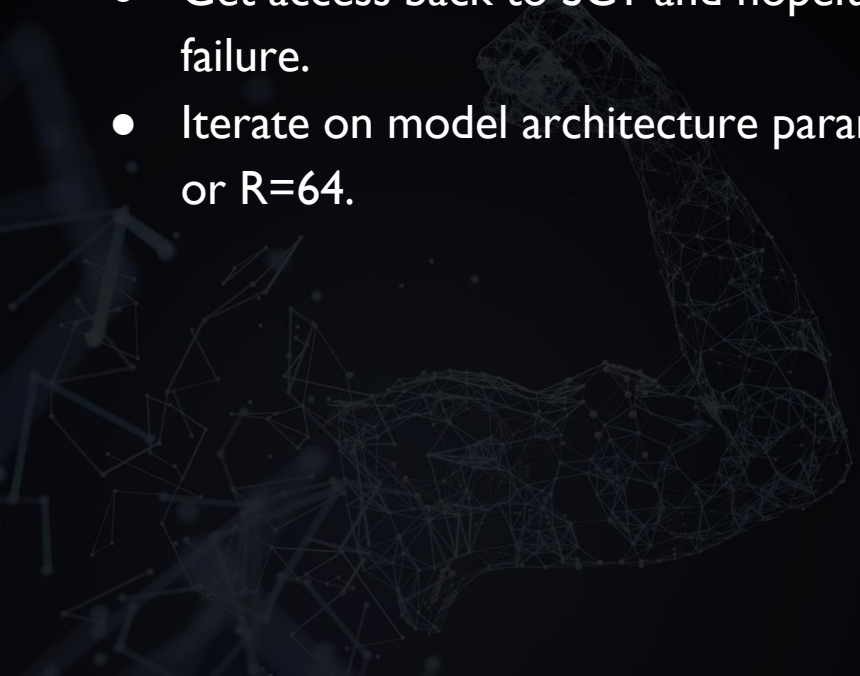
- Need to get SSH access back to SC1 and diagnose the failure.

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Get access back to SC1 and hopefully recover results of job running before SC1 failure.
- Iterate on model architecture parameterisation and re-run NAS, potentially at R=32 or R=64.



Strong Compute.


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
10/10/22 | WEEK 11 | MEETING 1
Video URL: https://youtu.be/EoNqTT7GC-w

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- ~~Draft~~ Progress Report Submitted  :
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Final Report to commence soon…

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Get access back to SC1 and hopefully recover results of job running before SC1 failure.	100%	Access back to SC1, results recovered.	
Iterate on model architecture parameterisation and re-run NAS, potentially at R=32 or R=64.	100%	NAS at R=64 complete and scaled model trained.	Results competitive with benchmarks!
Complete and submit progress report.	100%	Done.	Next stop final report.

Strong Compute.

Questions

- Yet to dive into the mixed precision methods with apex, if there are any working code examples you could share that would be awesome?

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- NAS on iterated search space, target to outperform benchmark;
  - removed SE operation from Fused-MBConv layers and introduced a fixed ordinary 2-stride convolution layer as model ‘head’ (per EfficientNet architecture).
  - Introduced parameterisation of expansion and SE bottleneck ratios.
  - Relaxed parameterisation of stochastic depth likelihood.
- Create draft Final Report document


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
13/10/22 | WEEK 11 | MEETING 2
Video URL: https://youtu.be/ln4b2OhOadM

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- ~~Draft~~ Progress Report Submitted  :
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Final Report document created
  - https://docs.google.com/document/d/1oWmbIYGI4BI2bf28PhcdAZBfxUOa9SOF/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
NAS on iterated search space, target to outperform benchmark	100%	Iteration on NAS ongoing.	A couple more adjustments made to bring architecture more into alignment with EffNet.
Create draft Final Report document	100%	Document created.	Yet to commence write-up in earnest.

Questions

- Still yet to dive into the mixed precision methods with apex. I did see the tutorial you shared Chris, so I'll hopefully get to dive into that this weekend.

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- NAS on iterated search space, target to outperform benchmark;
  - Added final Conv1x1 layer prior to classifier
  - Set bias=False for all Conv operations
- Implement mixed precision for speedup
- Populate ready-to-go content (e.g. lit review) in final report document


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

17/10/22 | WEEK 12 | MEETING 1

Video URL: https://youtu.be/3Uo2IP8UR5Y

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- ~~Draft~~ Progress Report Submitted  :
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Final Presentation document created (due Sunday November 13)
  - https://docs.google.com/presentation/d/1aDpPxcmrnZFVDFNs-aSu5jIJdju-JMml4ajdUjeQ_c4/edit#slide=id.p
- Draft Final Report document created (due Sunday November 20)
  - https://docs.google.com/document/d/1oWmbIYGI4BI2bf28PhcdAZBfxUOa9SOF/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Strong Compute.

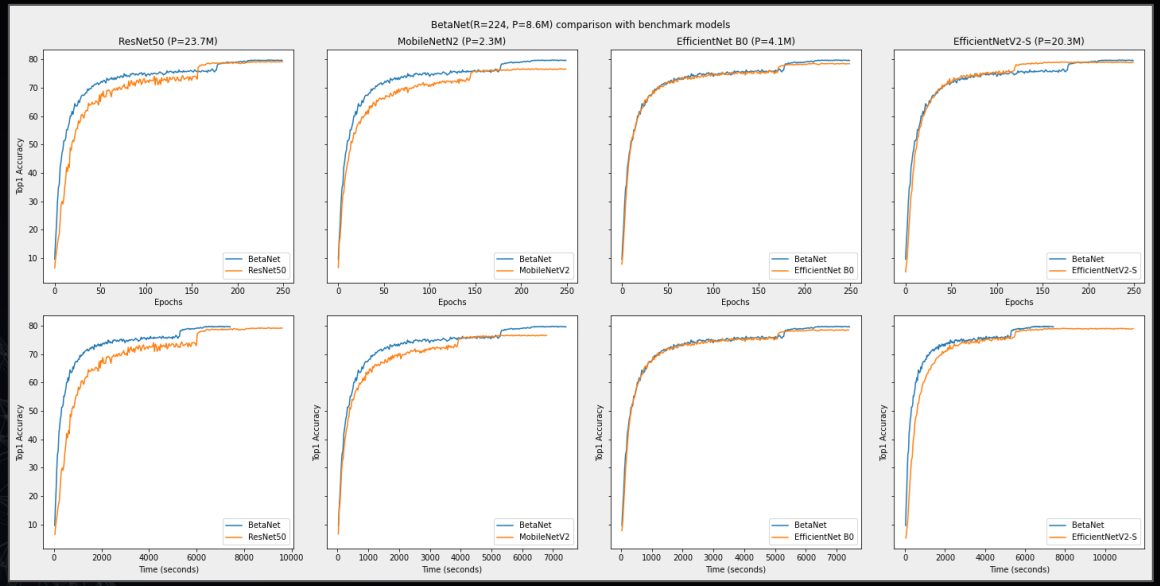
Progress

Aims From Last Meeting	How Complete	What was complete	Notes
NAS on iterated search space, target to outperform benchmark	100%	Iteration on NAS ongoing.	Additional parameters added to decouple nchannels from resolution. Results pending.
Populate ready-to-go content (e.g. lit review) in final report document	50%	Prepare final Presentation template.	Presentation due Sunday November 13.
Implement mixed precision for speedup	0%	Prioritised NAS search space iteration.	

Strong Compute.

Interim results

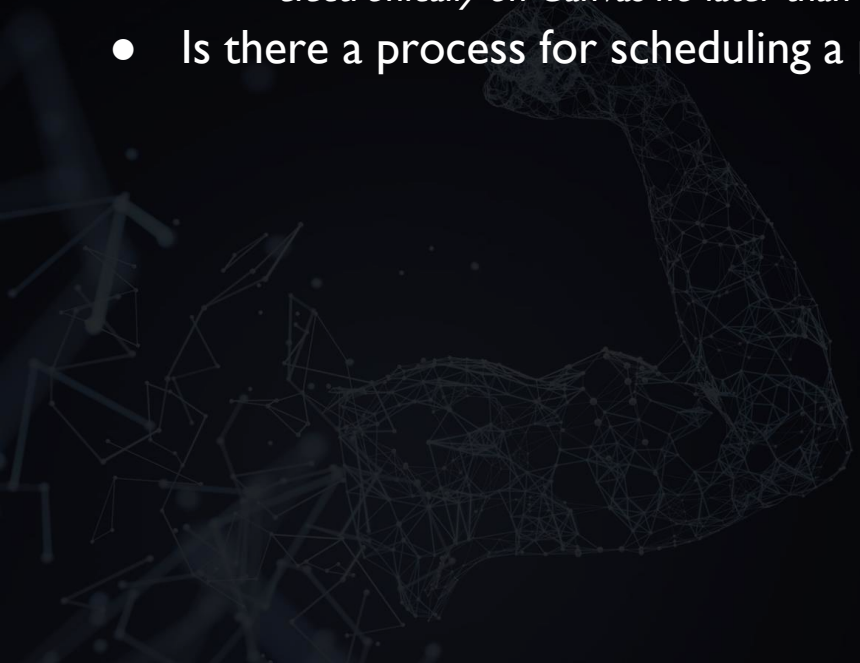
- BetaNet NAS results returned this model kernel which outperforms all benchmarks on CIFAR100.
- Faster to train than ResNet50 and EfficientNetV2.



Strong Compute.

Questions

- Instruction on USYD student portal:
  - *Please contact your supervisor to arrange a presentation time and submit your presentation slides electronically on Canvas no later than this due.*
- Is there a process for scheduling a presentation time with Strong Compute?



Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Obtain final NAS results
- Implement mixed precision for speedup
- Apply best BetaNet model found to ImageNet
- Start work on presentation and final report


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
20/10/22 | WEEK 12 | MEETING 2
Video URL: https://youtu.be/tycJpoUu0N4

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- ~~Draft~~ Progress Report Submitted  :
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Final Presentation document created (due Sunday November 13)
  - https://docs.google.com/presentation/d/1aDpPxcmrnZFVDFNs-aSu5jIJdju-JMml4ajdUjeQ_c4/edit?usp=sharing
- Draft Final Report document created (due Sunday November 20)
  - https://docs.google.com/document/d/1oWmbIYGI4BI2bf28PhcdAZBfxUOa9SOF/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

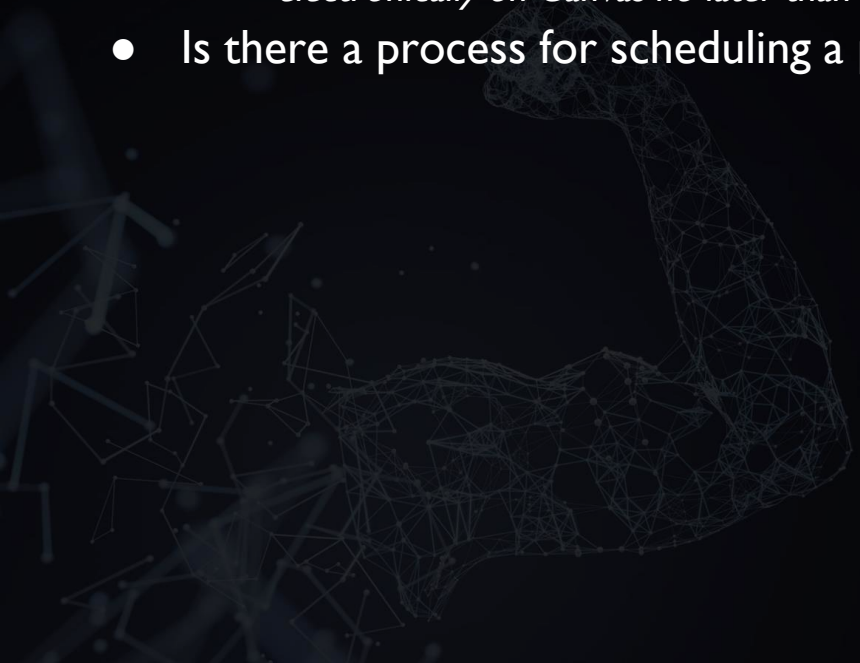
Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Obtain final NAS results	75%	Final NAS ongoing.	Additional parameters added to decouple nchannels from resolution. Results pending.
Implement mixed precision for speedup	0%	Prioritised final NAS this week.	Each run takes ~2 days to complete.
Apply best BetaNet model found to ImageNet	0%	Final NAS ongoing.	Fingers crossed for strong results to then apply to ImageNet.
Start work on presentation and final report	5%	Prepare final Presentation template.	Presentation due Sunday November 13.

Strong Compute.

Questions

- Instruction on USYD student portal:
  - *Please contact your supervisor to arrange a presentation time and submit your presentation slides electronically on Canvas no later than this due.*
- Is there a process for scheduling a presentation time with Strong Compute?



Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Obtain final NAS results
- Implement mixed precision for speedup
- Apply best BetaNet model found to ImageNet
- Start work on presentation and final report


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
24/10/22 | WEEK 13 | MEETING 1
Video URL: https://youtu.be/1qBd9KX5reQ

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- ~~Draft~~ Progress Report Submitted  :
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Final Presentation document created (due Sunday November 13)
  - https://docs.google.com/presentation/d/1aDpPxcmrnZFVDFNs-aSu5jIJdju-JMml4ajdUjeQ_c4/edit?usp=sharing
- Draft Final Report document created (due Sunday November 20)
  - https://docs.google.com/document/d/1oWmbIYGI4BI2bf28PhcdAZBfxUOa9SOF/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Obtain final NAS results	95%	NAS of last search space completed, training of scaled up model almost complete.	Contemplating approach to train / val / test.
Implement mixed precision for speedup	0%	Prioritised final NAS this weekend.	Each run takes ~2 days to complete.
Apply best BetaNet model found to ImageNet	0%	May supplant with alternative pending train / test / val decision.	
Start work on presentation and final report	15%	~50% of presentation drafted.	Presentation due Sunday November 13.

Questions

- None!

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Decide on and implement train / val / test approach.
- Implement mixed precision for speedup
- Apply best BetaNet model found to ImageNet or other dataset
- Continue working on presentation and final report


Strong Compute.
– END OF WEEK SLIDE –

USYD-17A

EfficientNetV2 Implementation

Strong Compute.

27/10/22 | WEEK 13 | MEETING 2

Video URL: https://youtu.be/yVjMHBNHcgI

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

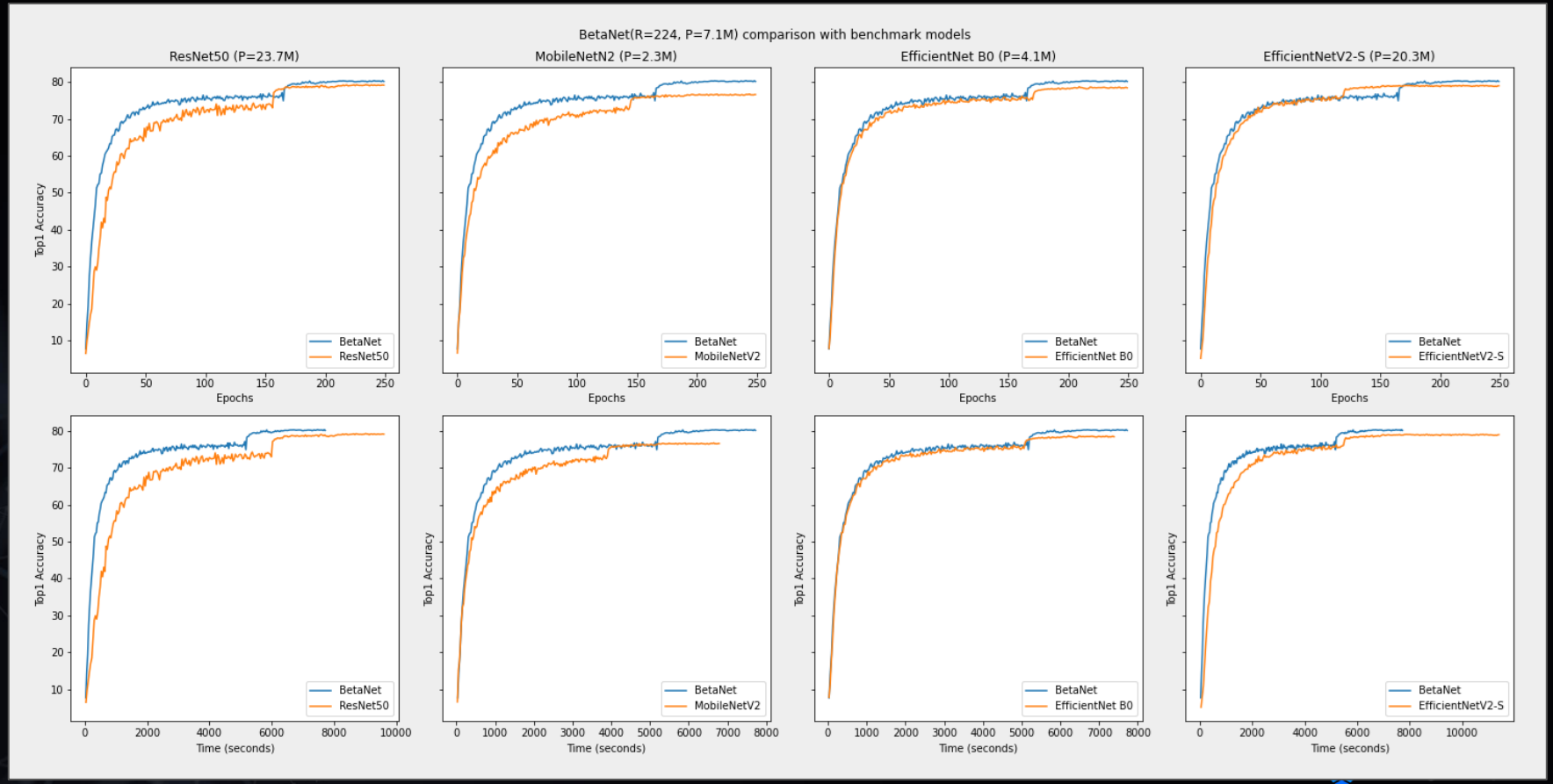
- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- ~~Draft~~ Progress Report Submitted  :
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Final Presentation document created (due Sunday November 13)
  - https://docs.google.com/presentation/d/1aDpPxcmrnZFVDFNs-aSu5jIJdju-JMml4ajdUjeQ_c4/edit?usp=sharing
- Draft Final Report document created (due Sunday November 20)
  - https://docs.google.com/document/d/1oWmbIYGI4BI2bf28PhcdAZBfxUOa9SOF/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Obtain final NAS results	100%	Applied NAS to R=224 directly, taking time constraint as definition of proxy task.	Strongest results yet. Re-framing of correspondence with EfficientNet scaling procedure.
Apply best BetaNet model found to Caltech256	10%	Obtained Caltech256 and split train into train/val.	Need to re-train all benchmarks on caltech256.
Start work on presentation and final report	20%	Incremental progress made on presentation.	Presentation due Sunday November 13.

Progress


BetaNet(R=224, P=7.1M) comparison with benchmark models
ResNet50 (P=23.7M)
MobileNetN2 (P=2.3M)
EfficientNet B0 (P=4.1M)
EfficientNetV2-S (P=20.3M)
Top1 Accuracy
Epochs
Time (seconds)
BetaNet
ResNet50
MobileNetV2
EfficientNet B0
EfficientNetV2-S

Questions

- None!

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Re-train benchmark models on Caltech256
- Apply best BetaNet model(s) found to Caltech256
- Continue working on presentation and final report

Strong Compute.


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
31/10/22 | WEEK 14 | MEETING 1
Video URL: https://youtu.be/mc_-RzfFiWE

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

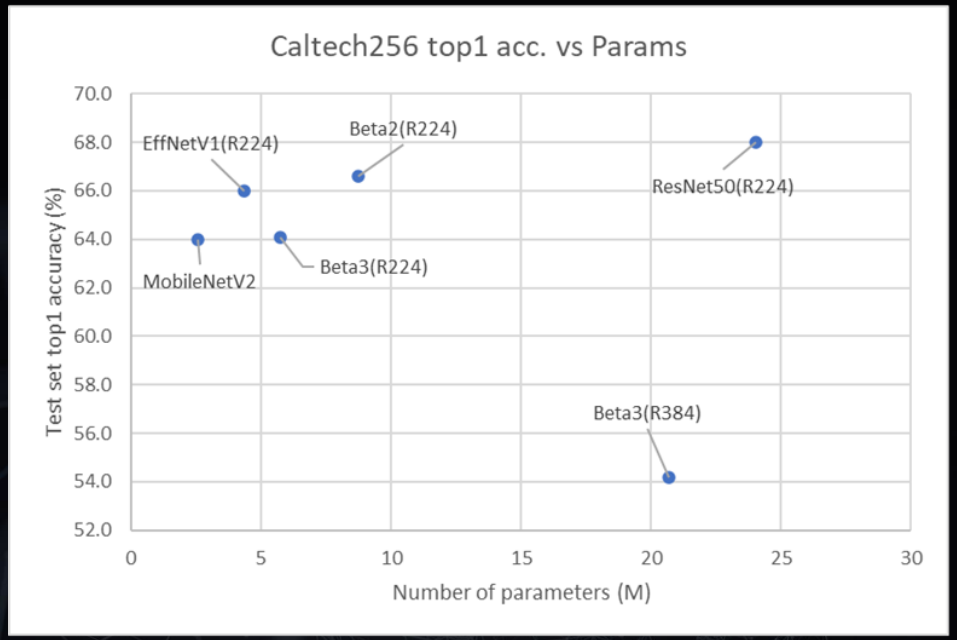
- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- ~~Draft~~ Progress Report Submitted  :
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Final Presentation document created (due Sunday November 13)
  - https://docs.google.com/presentation/d/1aDpPxcmrnZFVDFNs-aSu5jIJdju-JMml4ajdUjeQ_c4/edit?usp=sharing
- Draft Final Report document created (due Sunday November 20)
  - https://docs.google.com/document/d/1oWmbIYGI4BI2bf28PhcdAZBfxUOa9SOF/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

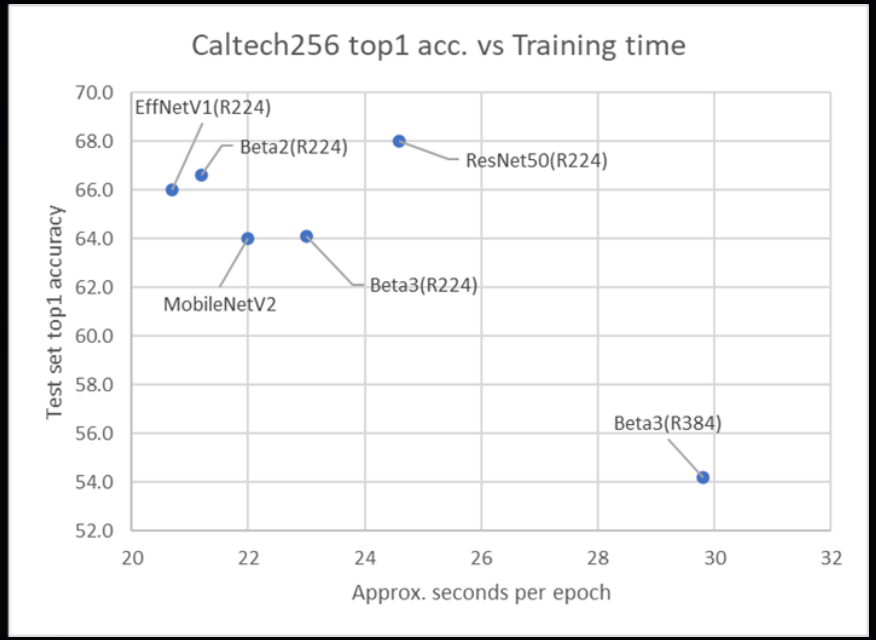
Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Progress

Aims From Last Meeting	How Complete	What was complete	Notes
Obtain final NAS results	100%	Applied NAS to R=224 directly, taking time constraint as definition of proxy task.	Strongest results yet. Re-framing of correspondence with EfficientNet scaling procedure.
Apply best BetaNet model found to Caltech256	95%	Obtained evaluation results for benchmarks, BetaNet2 and BetaNet3.	Mixed performance. BetaNet2 superior to BetaNet3 (and all benchmarks but ResNet50).
Start work on presentation and final report	50%	Report document assembled and populated with literature review and methodology content.	Presentation due Sunday November 13.

Progress





Strong Compute.

Questions

- Issue with Caltech256 that both BetaNets and EfficientNetV2 (R384) bombs on this dataset. Potentially due to small dataset size, but then early stopping does not seem to improve. Baffled.
- Considering abandoning Caltech256 and re-running BetaNet2 and BetaNet3 NAS on CIFAR100 split into train / val / test. NAS on train / val, then evaluate benchmarks on test.
- Not a huge amount of time left to get this done. Is there any chance I could have access to a second cluster to run benchmarks in parallel?
- Can we please schedule my presentation for Friday October 11 at 4pm?

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Subject to your feedback on plan and timing…
  - Re-run NAS on CIFAR100 split into train / val / test.
  - Re-evaluate benchmarks on CIFAR100
  - Get presentation and final report draft finished


Strong Compute.
– END OF WEEK SLIDE –


USYD-17A
EfficientNetV2 Implementation
Strong Compute.
31/10/22 | WEEK 15 | MEETING 1
Video URL: https://youtu.be/gQqMMSSRJHA

Any Resources/Sheets/Tables Docs etc.

If you created any resources, slides, documents, research, etc. please list them here:

- ~~Draft~~ Proposal Report Submitted  :
  - https://docs.google.com/document/d/1ZvpuhoiVigBvmzHgu9mjy-p_T21_mltl/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- ~~Draft~~ Progress Report Submitted  :
  - https://docs.google.com/document/d/1mZAMI_UYHnVKn4vDiMaeKORPWC8owUaS/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true
- Draft Final Presentation document created (due Sunday November 13)
  - https://docs.google.com/presentation/d/1aDpPxcmrnZFVDFNs-aSu5jIJdju-JMml4ajdUjeQ_c4/edit?usp=sharing
- Draft Final Report document created (due Sunday November 20)
  - https://docs.google.com/document/d/1oWmbIYGI4BI2bf28PhcdAZBfxUOa9SOF/edit?usp=sharing&ouid=103240485714029858431&rtpof=true&sd=true

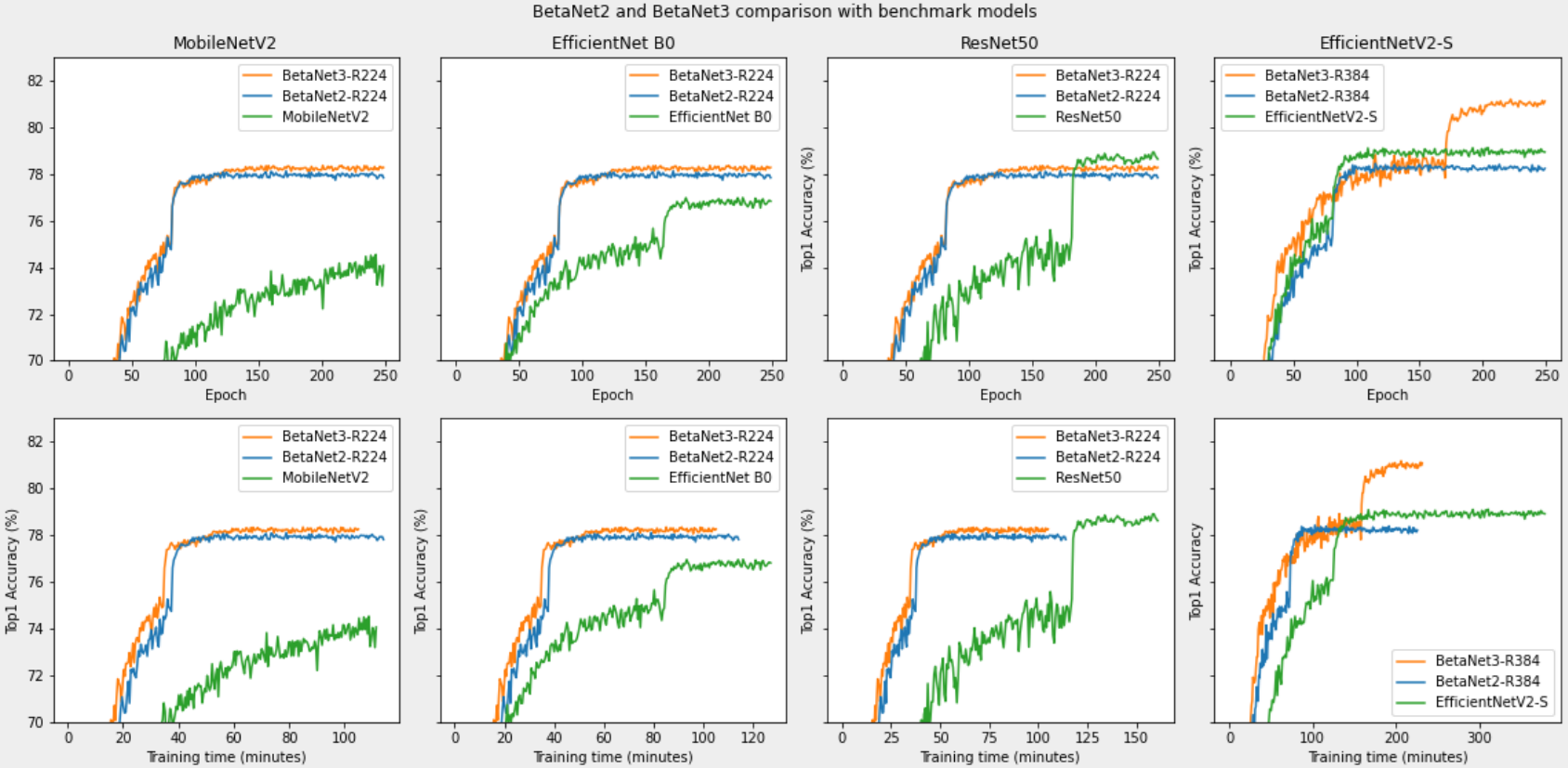
Please also message these links to us in the Discord channel on the morning of the meetings, by 10am.

Progress

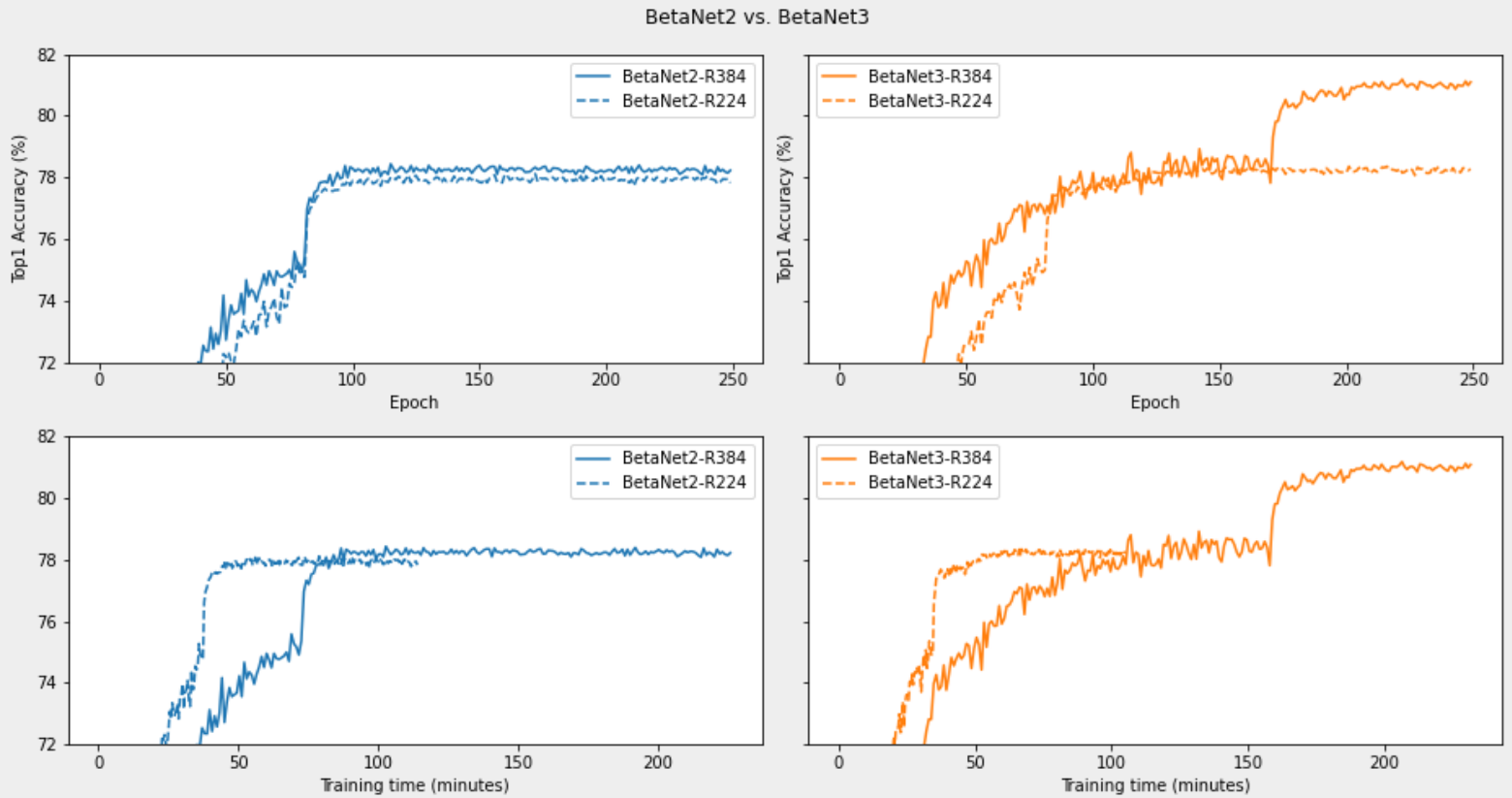
Aims From Last Meeting	How Complete	What was complete	Notes
Obtain final NAS and benchmark results	100%	All results obtained and visualised.	See following slides.
Start work on presentation and final report	80%	Report 80% drafted, presentation 80% complete.	Presentation scheduled for Thursday 10 November.

Strong Compute.

Progress


BetaNet2 and BetaNet3 comparison with benchmark models
MobileNetV2
EfficientNet B0
ResNet50
EfficientNetV2-S
BetaNet3-R224
BetaNet2-R224
MobileNetV2
EfficientNet B0
ResNet50
BetaNet3-R384
BetaNet2-R384
EfficientNetV2-S
Top1 Accuracy (%)
Epoch
Training time (minutes)

Progress


BetaNet2 vs. BetaNet3
BetaNet2-R384
BetaNet2-R224
BetaNet3-R384
BetaNet3-R224
Top1 Accuracy (%)
Epoch
Training time (minutes)
82
80
78
76
74
72
0
50
100
150
200
250

Questions

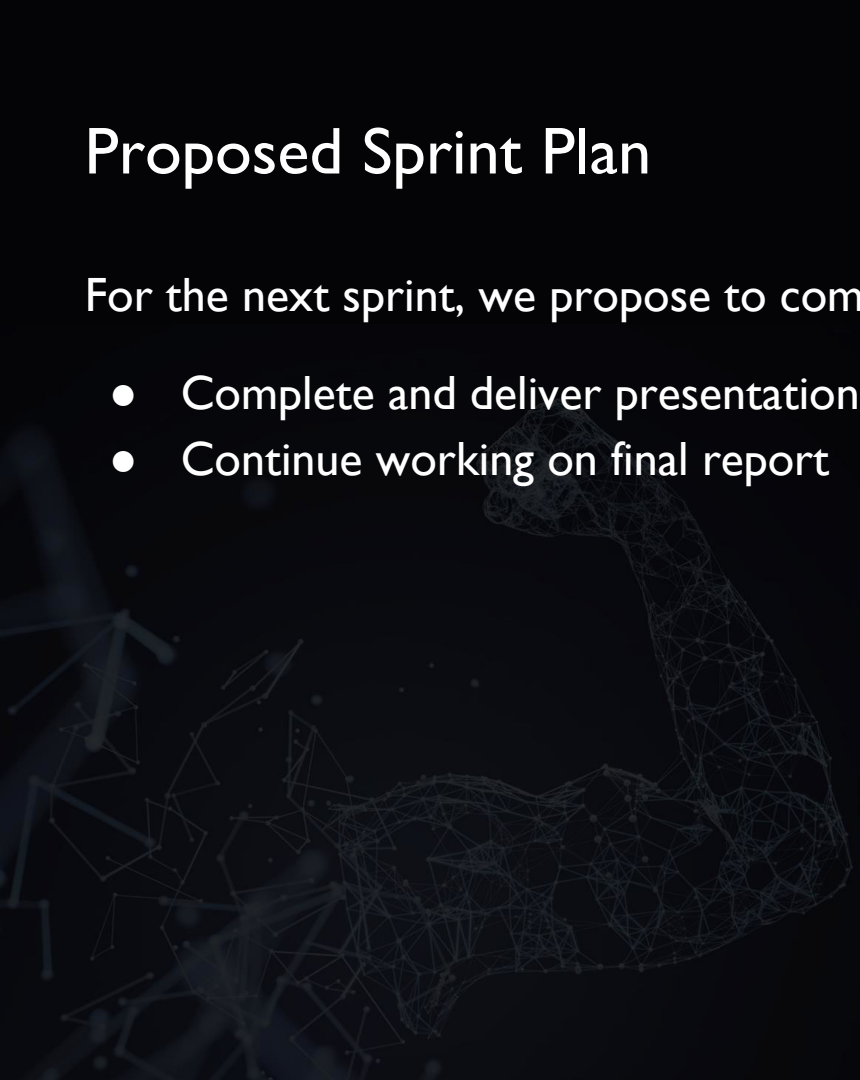
- Are there any specific requirements for project artefacts? Bundled and submitted in particular format or via particular channel?

Strong Compute.

Proposed Sprint Plan

For the next sprint, we propose to complete the following tasks:

- Complete and deliver presentation! (for Thursday)
- Continue working on final report



Strong Compute.


Strong Compute.
– END OF WEEK SLIDE –