MobileViT on TinyImageNet

Jeffrey B. Faath
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
jfaath3@gatech.edu

Avinash V. Uppuluri
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
auppuluri3@gatech.edu

Abstract

The MobileViT image classification network was designed as a lightweight and low latency model for mobile vision tasks. MovileViT employs standard convolutions layers taking advantage of their inductive bias towards vision-related tasks and vision transformers to incorporate global information via its self-attention mechanism. By combining these paradigms, MobileViT is able to produce comparable results to much larger networks on the ImageNet-1k data set. We propose an offshoot model, MobileViT-Tiny, that is customized to work with the TinyImageNet data set, a highly compressed and stripped down version of ImageNet-1k. We attempt to produce the best results possible by modifying the original MobileViT network and compare the results to the well-known ResNet model.

1. Introduction

Ten years ago, AlexNet [7] took the computer vision world by storm by vastly outperforming the competition in the 2012 ILSVR Challenge. CNNs have since been refined, becoming more efficient and even adapted to work well with the more computationally challenged mobile devices. Five years ago, the Transformer [10] burst onto the scene with the concept of self-attention, breaking records in the natural language processing (NLP) space. Thanks to the transformer's success in NLP, researchers sought to adapt the architecture to computer vision tasks. Thus, the Vision Transformer [6] was born facilitating representations in the global image space (as opposed to local a la CNNs). Transformer architectures, however, are heavyweight and not necessarily conducive to mobile devices ubiquitous in today's world. The authors of "MobileViT: Light-Weight, General-Purpose, and Mobile-Friendly Vision Transformer" [8] tackle this problem by introducing an architecture that combines both CNNs and Vision Transformers to "build a light-weight and low latency network for mobile vision tasks".

In this study, we attempt to re-evaluate the author's work from MobileViT to determine if top performance can in fact be attained in a computationally efficient manner through the evolution and adaptation of historical discoveries in the deep learning space. The authors evaluated their model based on the ImageNet-1k [3] data set, which contains nearly 1.3 million images in 1,000 categories. Due to computational and time constraints we will re-evaluate the results based on a smaller subset of ImageNet – TinyImageNet [4, 2].

Images in TinyImageNet are a quarter of the size of standard ImageNet-1k images and importantly, are actual shrunken views of the original images. This means object features are equally compressed as no attempt was made to extract salient parts of the original images and simply place them on a 64x64 canvas. Additionally, TinyImageNet only offers 100,000 images versus 1.28M for ImageNet-1k. These images are distributed over 200 categories for 500 exactly per category whereas ImageNet-1k, though it contains 1,000 categories, has on average over twice as many images per category. The validation set contains 10,000 with 50 per class. This set is used as the benchmark in all experiments as TinyImageNet does not provide a labeled test set.

As far as we know, no one has attempted to use the TinyImageNet dataset on MobileViT. Achieving strong results on this data set will further validate MobileViT as a viable lightweight image classification architecture.

2. Approach

We will attempt to produce the best results possible on TinyImageNet with the MobileViT architecture. First we will adapt MobileViT for the new data set on a superficial level to produce a performance baseline. From there we will tune hyperparameters and the architecture itself in hopes of achieving comparable results to the MobileViT authors. For comparison purposes, we will follow the same procedure on the well-known purely-CNN ResNet architecture, starting with three common variations as a baseline and improving from there. Our initial concerns are potential for overfitting since these models have capacities designed for more extensive data sets. We hope that our experiments will discover

and correct any such issues.

The authors provide full source code [1] for MobileViT which includes a configurable implementation of ResNet. This code was branched for purposes of this study. Both models required common adaptations to be compatible with TinyImageNet with the first obvious step being to reduce the output of the classification layer to the smaller class amount. The authors used label smoothing and we chose to reduce the label smoothing constant by a factor of 5 to reflect the reduction in classes. Finally, the authors employed variable batch sampling which varies the batch size and image scale through training. Like ResNet and other CNNs, the MobileViT architecture is agnostic to input shape thanks to its initial use of CNN layers (unlike pure ViT models which typically train on different scales independently due to the use of positional embeddings). The authors worked with 5 different scales each 32 pixels apart both below and above the original image scale. Due to the much smaller size of TinyImageNet images, we chose 5 scales varying by 6 pixels with only 1 variant below the original size and the rest above.

2.1. MobileViT

With the adaptations in place, the MobileViT model was trained using the TinyImageNet data. One additional change - the model was trained for 200 epochs versus the original 300 used on ImageNet-1k since TinyImageNet contains far fewer data. The authors provide three flavors of their model differing only by the size of the output channels and feed forward network nodes. These are small, x_small and xx_small, listed from most parameters to least. As MobileViT was designed for portable devices, the authors prioritized building models with fewer parameters (which certainly had an influence on the model naming scheme!). Table 1 shows the results for all three model flavors in terms of accuracy.

model	train top1	**val top1**	val top5	# params
small	54.03	**44.91**	70.37	5.6M
x_small	55.97	***53.42***	78.51	2.3M
xx_small	52.14	**50.36**	75.75	1.3M

Table 1. MobileViT baseline runs

All three models flavors can be seen to overfit by a magnitude that corresponds to their number of parameters. Interestingly, the middle model performs the best on the validation set. Though it overfits slightly, the capacity of the x_small model seems best aligned for the data set with the xx_small a somewhat distant second. The largest model has the most parameters by far and produces significantly worse baseline results. Clearly the reduced feature space borne out of the compressed images is better serviced by smaller model capacity. Further tuning will consider this fact.

A brief summary of the MobileViT architecture was helpful in informing further model optimizations. At a high level, MobileViT is composed of an initial downsampling convolution layer, five component layers and a final classification layer. The first two component layers, called MV2 by the authors, consist of a configured amount of inverted residuals blocks introduced by MobileNetV2 [9]. These blocks offer efficient mobile-friendly convolutions with residuals and are thus designed to capture localized features while also downsampling the input image depending on the configured stride. More information can be found in [9] and will not be discussed further. The next three components layers are called MobileViT layers. Each of these layers start with a downsampling inverted residual followed by a MobileViT block - a chief contribution of the author's paper. The MobileViT block consists of a 3x3 convolution to capture local information, Transformer blocks to discover global connections and a final convolution to fuse the results with the input to the block. Of interest is the MobileViT block's unique method for capturing both local and global information. Figure 1 shows the attention mechanism that performs this function.

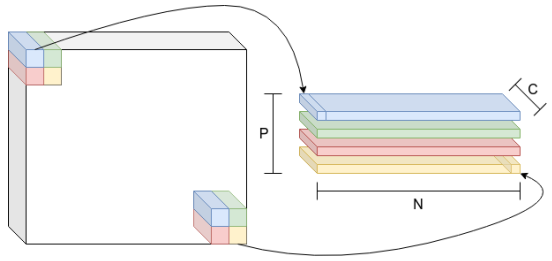

Figure 1. MobileViT attention mechanism.

The left side of the figure represents input to the MobileViT block after the initial convolution (and a pointwise convolution to expand depth). Each pixel represents local information owing to the prior 3x3 convolution. The input is then split into separate 2x2 patches (as shown in the figure) through which attention is applied. The trick is each position in the patch along with its channel depth is rolled into the batch dimension producing P=4 rows per sample. Each row represents all the pixels of the given patch position (all the first pixels, all the second pixels, etc). Self-attention is applied to each of these rows so that each patch pixel attends to all other same patch pixel positions - not unlike a dilated convolution. As each patch pixel represents local information courtesy of the convolution, the self-attention combines that local information with the global space. The number of patches N plays an important role here and that value depends on the shape of the input after downsampling. Armed with this architectural information we set out to modify Mo-

bileViT to suit the TinyImageNet data set.

2.2. ResNet

A comparative analysis is provided against ResNets of different depths. Table 2 provides the number of parameters and base line performance of ResNets we experimented with. Even the lowest depth ResNet experimented with (i.e. ResNet-18) has about twice as many parameters as the largest MobileViT configuration (i.e. MobileViT-small).

We started off with the understanding that a smaller configuration like ResNet-18 will be more appropriate for the TinyImageNet dataset given its size. The intent of experimenting with the different depths of ResNet was to find out if including more parameters and applying ways to prevent overfitting help obtain better performance on the TinyImageNet dataset. From the baseline numbers we picked ResNet-50 as an additional model to pursue further given its slightly better performance than ResNet-18.

Since we spent considerable effort fine tuning MobileViT to attain best possible performance we also did similar fine tuning of the ResNet models to keep the comparison fair across models. The goal of fine tuning the ResNet models was to obtain the best possible performance on the TinyImageNet data set.

The standard ResNet architecture with 18 layers and residual connection within the basic ResNet blocks is shown in Figure 5. The parameters of the baseline models and further fine tuning experiments are explained in section 3.2.

model	train top1	**val top1**	val top5	# params
ResNet-50	54.61	***55.70***	80.18	23.9M
ResNet-34	50.62	**52.78**	76.74	21.4M
ResNet-18	53.84	**54.37**	78.26	11.3M

Table 2. ResNet baseline runs

3. Experiments and Results

With both models adapted for use on TinyImageNet, baseline runs in place, and a proper understanding of the inner workings of each model, we set out to produce the best results via experimentation. This process and the ensuing results will be detailed in this section.

3.1. MobileViT

Given the results from the baseline runs, where the x_small model outperformed the other two model flavors, our first decision was to focus on the x_small variant. We did initially attempt to run experiments that were successful with x_small on the other two flavors, however, these runs consistently came in below the x_small result to the point where we stopped using the other variants altogether. Thus, all experiments in this section can be assumed to be based on x_small. Additionally, all experiments were trained for 200 epochs and used the default cosine learning rate scheduler and warmup value as well as the Adam optimizer as some preliminary attempts to tune these values were unsuccessful. The summary of the experimental results are found in Table 3 and will be discussed in detail here.

experiment	train top1	**val top1**	val top5
no variable batch	58.69	**57.29**	80.49
remove 5	54.69	**57.81**	81.39
remove 4,5	58.53	**58.73**	81.86
remove 4,5 decrease L2	54.98	**56.73**	80.95
remove 3,4	61.30	**59.67**	82.36
remove 3,4 no fusing	60.67	***60.11***	82.49
remove 3,4 use residual	59.98	**59.76**	82.00
remove 3,4 no fusing, cutout	58.29	**59.63**	82.50

Table 3. MobileViT experimental runs using x_small base

As mentioned earlier, the authors used variable batch sampling when training on ImageNet-1k and we altered this procedure to accommodate the smaller images of TinyImageNet for the baseline runs. While this training method is intended to aid the model in learning multi-scale representations, we decided that with the already heavily scaled down images in our data set, varying the input sizes would not be useful - particularly since TinyImageNet images are 64x64 giving little leeway to scale downward from this starting resolution. Indeed, removing the variable batch sampling provided a significant accuracy boost over the baselines as seen in the first row of Table 3. All future experiments therefore removed variable batch sampling. On the topic of data augmentation, the original MobileViT employed only two transformations: random-resized-crop and horizontal flip. The authors state their model is robust to hyperparameter tuning such as data augmentation and we discovered this for ourselves as attempts to add augmentations such as random noise and cutout served only to reduce performance, albeit slightly. This was unfortunate since we believed TinyImageNet would benefit from augmented data given its small size. Nevertheless data augmentation was mostly discarded in future experiments.

We next delved deeper into the MobileViT architecture in search of improvement opportunities. As the original MobileViT model was designed for images four times the size of TinyImageNet, we thought to inspect the final output shape resulting from the smaller input images. Sure enough, we discovered that with the original downsampling in place, the final output of the encoder before the classification layer was 2x2 (versus 8x8 for ImageNet-1k). This

presented a problem since the image patches used in MobileViT's attention mechanism are in fact 2x2. Thus on the last MobileViT block only one patch would be extracted rendering the attention mechanism pointless! We decided to remove the last (fifth) layer entirely and saw a slight performance gain and possible underfitting (see Table 3) likely because the final layer is responsible for about 36% of the network capacity. We tested this theory by also removing the fourth, also a MobileViT layer, but to our surprise we observed another performance boost and good training behavior with similar training and validation accuracy. Upon further inspection, we realized with layers 4 and 5 removed the Transformer input shape of the only remaining MobileViT block was 8x8, the shape meant for the last encoder layer in the original model. The stronger performance indicates that learning global representations courtesy of a larger receptive field is beneficial as the authors intended. Given that layers 4 and 5 contained a large portion of the network capacity (roughly 67%), we thought reducing L2 weight decay, which was set at a relatively high 0.01 by default, might open the network up for learning better representations. This turned out to be an incorrect hypothesis as using 0.001 weight decay delivered a 2% hit to accuracy. We still felt the network would benefit from increased capacity and came upon a simple idea. Since leaving only one MobileViT block with a larger receptive field proved fruitful, why not leave the block with the most capacity? We had removed blocks 4 and 5 - the higher capacity blocks - so to test this idea, we instead removed blocks 3 and 4 leaving the highest capacity fifth block. We were rewarded with another percentage point gain in top-1 validation accuracy.

The MobileViT block is a self-contained unit where the input shape is equal to the output shape. The authors chose to add a residual-like strategy where they concatenate the output of the block with the input along the depth dimension and then run a convolution to "fuse" the input and output returning the original depth. Presumably this is to ensure better gradient flow by learning only relevant linear combinations of the input with the output. This strategy likely made sense with multiple MovileViT blocks but our modified model is left with only one of these blocks. We decided to remove this fusing operation and received another accuracy boost pushing the result over 60% for the first time.

Further experiments included attempting to use a standard residual strategy (adding the input to the output and normalizing) instead of the fusing and reintroducing cutout on our best result, but both ideas degraded results slightly. Our final model, which we dubbed MobileViT-Tiny, achieved a top-1 accuracy of 60.11% and a top-5 accuracy of 82.49% on the validation set (Table 3). The architecture of MobileViT-Tiny including its comparison with the original MobileViT architecture can be seen in Figure 2.

3.2. ResNet

We trained the baseline ResNet models with batch sizes of 128. Basic data augmentation was done using random resized crop and horizontal flip. A base Stochastic Gradient Descent optimizer was used with a weight decay of 1.e-4. A cosine scheduler was used to vary the learning rate during training with an initial warm-up to 0.4 over 7500 iterations. The experiments done on top of the base configuration with ResNet-18 and ResNet-50 models are provided in Table 4.

As mentioned in section 3.1, it did not make sense to allow variable input sizes that went below the already small size of TinyImageNet images. Hence the capability to use smaller image sizes with variable batches was disabled for our experiments with TinyImageNet. As shown in experiment B and G of Table 4, this provided a boost of 2.4 and 4.5 percent points in the case of ResNet-18 and ResNet-50, respectively.

Given that TinyImageNet had fewer samples per category compared to ImageNet, data augmentation methods were applied to check if they can help improve performance. It was observed that applying various augmentation techniques that decreased the active area of an already small TinyImageNet image proved detrimental to performance. Augmentations that provided distortions while preserving the image size seemed to fare slightly better. Related results are tabulated in the experiments C and E of Table 4 with the experiment tags *augment tiny*. Overall, going from experiment B to C and likewise from D to E show a decrease in performance of top-1 validation accuracy by about five percentage points. The smaller size of the TinyImageNet dataset may be a possible cause for the lack of validation performance improvement expected on data augmentation. Data augmentation did provide a way to prevent the early onset of over-fitting. As shown with loss values from experiment K in Fig. 3, the validation loss stays lower that training loss, when data augmentation methods are used. This trend wit lower validations loss is seen all the way through till the end of training, for 150 epochs.

The ResNet-18 and ResNet-50 models were experimented with different schedulers to change learning rate, to see their affects on model loss minimization and performance maximization. Three schedulers - cosine, cyclic and divide-by-ten were used in our experiments. While the first two were provided with the MobileViT repository, we implemented the divide-by-ten scheduler as it was referenced in ResNet related literature [5].

The cyclic scheduler, does a configurable number of high-to-low sweeps of the learning rate over a range of epochs. Experiments with this method did not prove very fruitful in terms of performance improvement. Our implementation of the divide-by-ten, as the name suggests, would divide the learning rate by ten when the performance of the model plateaued over a window as shown in Fig. 3. This

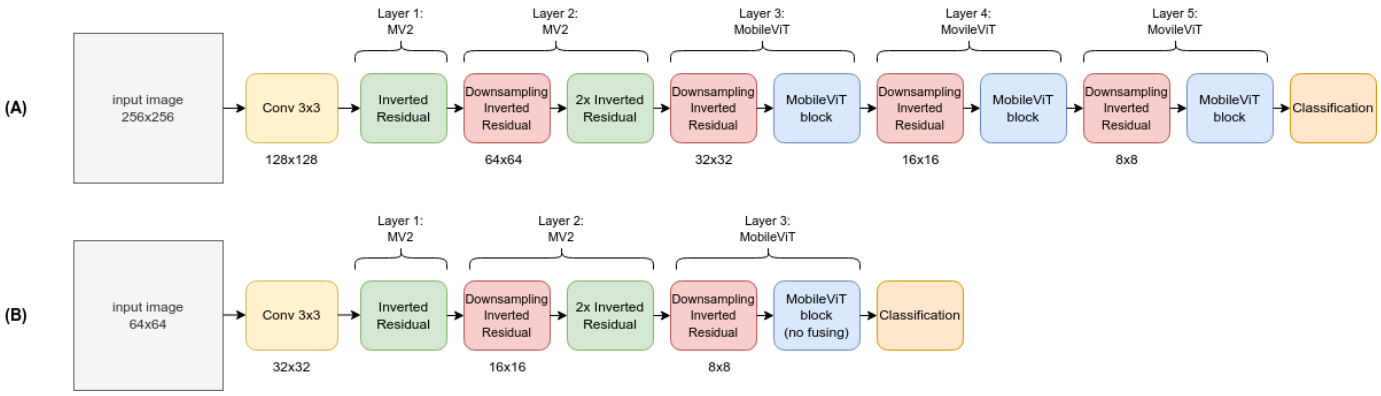

Figure 2. **(A)** The original MobileViT architecture, and **(B)** modified architecture of MobileViT-Tiny

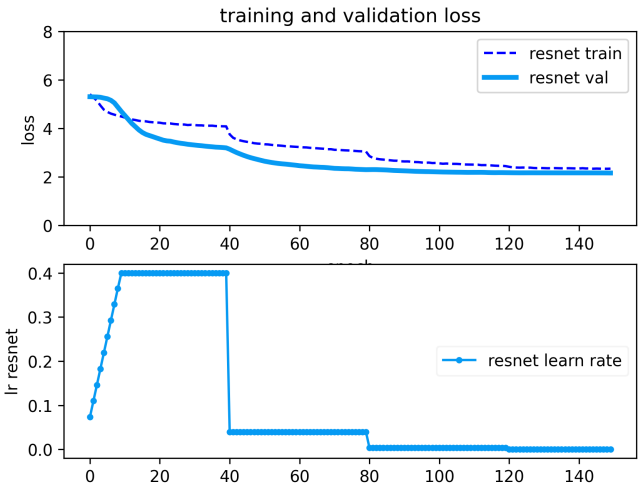

Figure 3. ResNet sample loss behaviour on top, with data augmentation and div-by-ten scheduler below.

method did reasonably well as listed in experiment D, when comparing with experiment B for cosine scheduler, of Table 4 for ResNet-18. The divide-by-ten scheduler does fall short by 1.4 percent points for ResNet-18 performance and 6.3 percent points for ResNet-50, when compared to the cosine scheduler. The cosine scheduler provided the best performance in most experiments (see Fig. 4). It gradually decreases from a maximum to a minimum learning rate over the number of epochs of training following a cosinusoidal curve.

Contrary to our assumptions that the small dataset size of TinyImageNet would do better, in terms of overfitting, with a model that has smaller number of parameters (i.e. ResNet-18) it turned out that the ResNet-50 model with variable image size inputs disabled (i.e., experiment G in Table 4), did the best with respect to top-1 and top-5 validation performance numbers.

#	experiment, scheduler	train top1	**val top1**	val top5
	ResNet-18			
A.	base, cosine	53.84	**54.37**	78.26
B.	no variable batch, cosine	54.97	***56.74***	79.70
C.	no variable batch, cosine augment tiny	44.80	**51.07**	77.00
D.	no variable batch, div-by-ten	54.56	**55.30**	78.37
E.	no variable batch, div-by-ten augment tiny	43.13	**49.78**	77.07
	ResNet-50			
F.	base, cosine	54.61	**55.70**	80.18
G.	no variable batch, cosine	58.51	***60.19***	81.93
H.	no variable batch, cosine augment tiny	47.67	**54.59**	79.11
I.	no variable batch, cosine weight decay	55.89	**57.22**	80.78
J.	no variable batch, div-by-ten	53.54	**53.89**	78.41
K.	no variable batch, div-by-ten augment tiny	47.27	**51.70**	78.30

Table 4. ResNet-18 and ResNet-50 experimental runs

4. Conclusion

Both our modified version of MobileViT and the standard implementation of ResNet-50 offered comparable results on the TinyImageNet data set. Figure 4 shows the top-1 training progression of the best runs for both models on the training and validation sets. Since the results top out at just over 60% for both models despite excessive tuning and tweaking, we are left to wonder if the images of TinyImageNet have an accuracy cap given their heavily compressed feature space. Nevertheless, several observations can be made. Both models achieve their best results somewhere in the 100 to 130 epoch range. MobileViT-Tiny jumps out to an early lead while ResNet-50 eventually catches up. We speculate that with ResNet-50's deep architecture and heavy use of residuals, training proceeds at a slower pace since

residuals learn differences which take longer to manifest across ResNet-50's many layers. While MobileViT-Tiny maintains some residual connections, they are not nearly as ubiquitous as ResNet-50 and MobileViT-Tiny's smaller architecture and use of Transformers makes it the more opinionated network and thus initially faster to learn. Indeed when ResNet-50 is left to continue training, it can be observed to overfit (Fig. 4) quite severely owing to much greater capacity and the fact that the residual layers have expanded. The smaller epochs values and overfitting (for ResNet-50) or stalled training (MobileViT-Tiny) lends credence to a possible upper bound on TinyImageNet's training and validation accuracy.

While both networks perform similarly, it should be noted that MobileViT-Tiny is a much smaller network compared to ResNet-50. In fact, MobileViT-Tiny has a mere 817k parameters compared to 23M parameters contained within ResNet-50. This is an important point and validates the author's intention for their MobileViT architecture. They achieved similar results on ImageNet-1k and we have shown that this holds true for a different data set with our modified version, MobileViT-Tiny. The combination of light-weight mobile friendly convolutions along with a self-attention mechanism to merge local information with the global space - courtesy of the MobileViT block - has great potential to provide commensurate accuracy of much more heavy-weight models under the constrained environment of mobile devices.

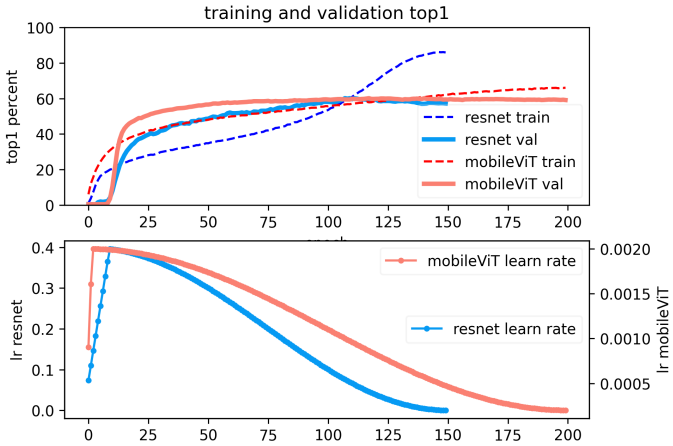

Figure 4. MobileViT and ResNet best performance.

5. Work Division

Contribution from team members is provided in Table 5.

References

[1] Apple. Mobilevit code. https://github.com/apple/ml-cvnets. Accessed: 2022-03-25. 2

[2] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A Downsampled Variant of ImageNet as an Alternative to the CIFAR datasets. *arXiv e-prints*, page arXiv:1707.08819, July 2017. 1

[3] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009. 1

[4] Li Fei-fei. Tiny imagenet dataset. http://cs231n.stanford.edu/tiny-imagenet-200.zip. Accessed: 2022-03-20. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4

[6] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021. 1

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 1

[8] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022. 1

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018. 2

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 1

A. Appendix

Student Name	Contributed Aspects	Details
Jeffrey B. Faath	MobileViT	Tuning and training MobileViT model on TinyImageNet
Avinash V. Uppuluri	ResNet	Tuning and training ResNet model on TinyImageNet

Table 5. Contributions of team members.

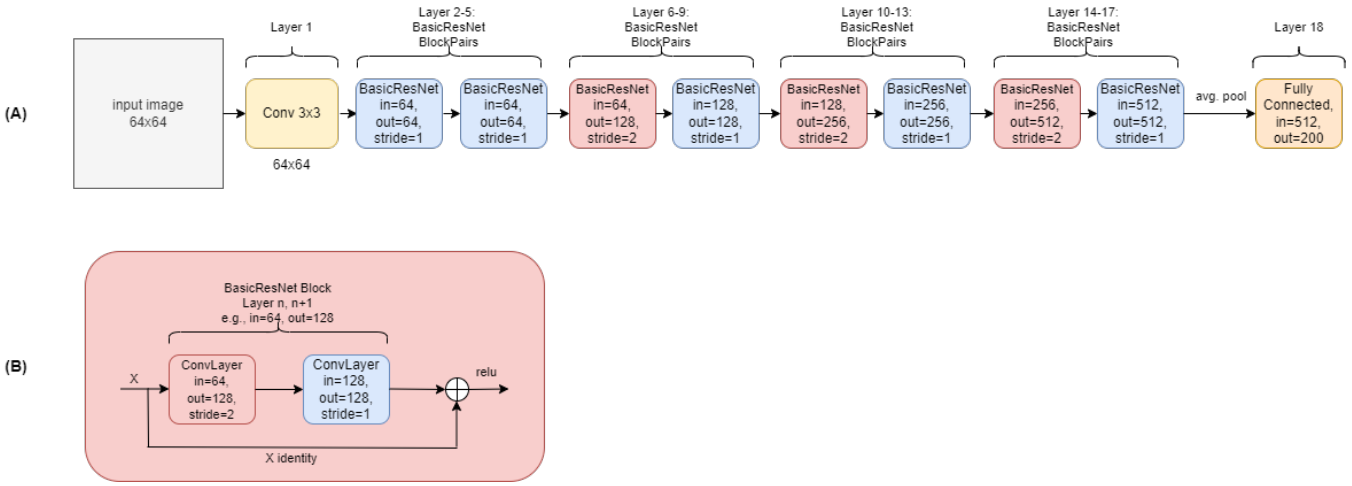

Figure 5. **(A)** ResNet-18 architecture, and **(B)** Basic ResNet Block with residual connection.

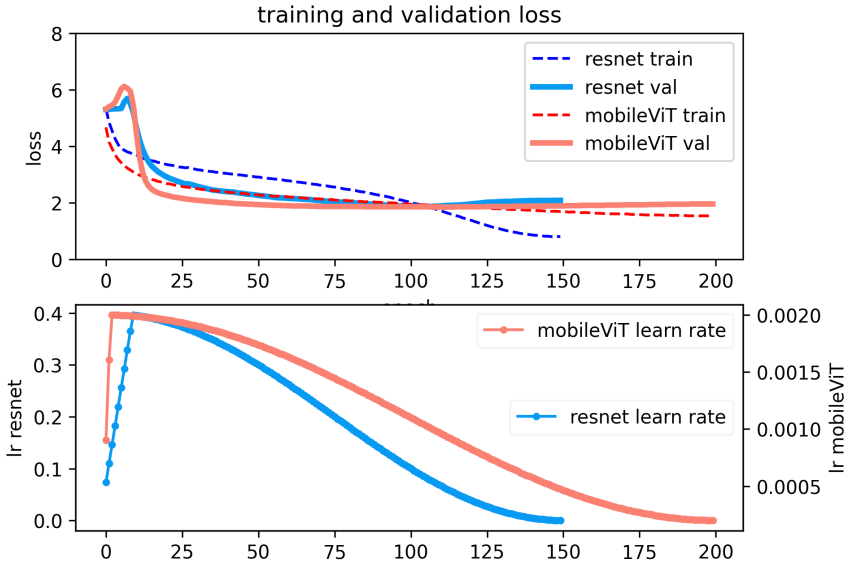

Figure 6. MobileViT and ResNet best performance.