50.035 Computer Vision Project Report
Compact Vision Transformers for Classification

Bryan Tan (Chen Zhengyu) | 1004318
Christy Lau Jin Yun | 1005330
Hung Chia-Yu | 1005603
Mandis Loh | 1005297
Jared Lim | 1005200

Computer Science and Design (CSD)
Singapore University of Technology and Design

December 15, 2023

Abstract

This study addresses the challenge of optimizing Compact Convolutional Transformers (CCT) for classification by improving their compactness and data efficiency. We applied various modifications to CCT models of varying complexities and evaluated them on the CIFAR-10 and CIFAR-100 datasets. These modifications include Simple Ensembling, Dynamic Positional Embedding (DynEmbed), Trainable Temperature-Scaled Attention, Low-Rank Projection of Linear Layers, and Squeeze-and-Excitation Blocks (SEBlocks) integration. Our findings reveal that Simple Ensembling offers limited benefits compared to its parameter increase while DynEmbed yielded a moderate accuracy improvement but at the cost of higher parameter counts. Trainable Temperature-Scaled Attention offered a slight accuracy boost with minimal additional parameters while the parameter reduction offered by our implementation of Low-Rank Projection is overshadowed by its decrease in accuracy. Lastly, adding SEBlocks resulted in a negligible change in accuracy, suggesting that their effectiveness may be context-dependent within CCT architectures. These results suggest that while some strategies like DynEmbed and Trainable Temperature-Scaled Attention show promise, improvements often come with trade-offs, and further optimization is required to achieve the goal of making Vision Transformers more compact and data-efficient. Overall, the study explored promising directions for CCT performance optimization, proposed reasons for any changes in performance and opened avenues for future exploration into alternative architectural changes. The code and instructions for running them can be found at https://github.com/Inc0mple/Compact-Transformers.

Contents

1 Introduction

Vision Transformers (ViT) have achieved impressive results for various Computer Vision tasks, often matching or surpassing traditional Convolutional Neural Networks (CNNs). However, they are large and data-hungry and known for their performance on large datasets rather than their efficiency on smaller datasets. Such models can thus be challenging to deploy in resource-constrained environments We seek to implement and advance existing implementations of compact vision transformers such as Compact Convolutional Transformers (CCT). Such work not only helps to lower model energy consumption, enable edge/mobile deployments and democratise State-of-the-Art Computer Vision techniques it can also advance Natural Language Processing (NLP) techniques and help unify Vision-Language modelling.

We hence aim to solve the following problem: Implement and improve the Compact Convolutional Transformer (CCT) architecture to:

1. Increase classification accuracy on the CIFAR-10 and CIFAR-100 datasets for a given parameter count, or...
2. Decrease parameter count while maintaining accuracy on the CIFAR-10 and CIFAR-100 datasets.

2 Dataset

2.1 CIFAR-10

The CIFAR-10 dataset contains 60000 RGB colour images of size 32x32 px, the images are grouped into 10 classes, with 6000 images per class. The dataset is evenly divided into 50000 training images and 10000 test images. The 10 classes represent common objects such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. CIFAR-10 serves as a benchmark for testing the performance of machine learning models in image recognition tasks, providing a challenging yet manageable dataset for researchers and practitioners to develop and evaluate their algorithms.

2.2 CIFAR-100

CIFAR-100 is an extension of CIFAR-10, also containing 60000 RGB colour images of size 32x32 px. However, the CIFAR-100 has 100 different classes, with 600 images per class. The classes are more diverse and fine-grained, covering a broader range of objects and concepts. The 100 classes are further categorised into 20 superclasses, each containing 5 sub-classes. This dataset is designed to be more complex than CIFAR-10, challenging models to learn and generalise across a larger set of categories. It serves as a valuable resource for testing the robustness and versatility of image classification algorithms in real-world scenarios.

3 Related Works

3.1 Convolutional Neural Networks (CNNs)

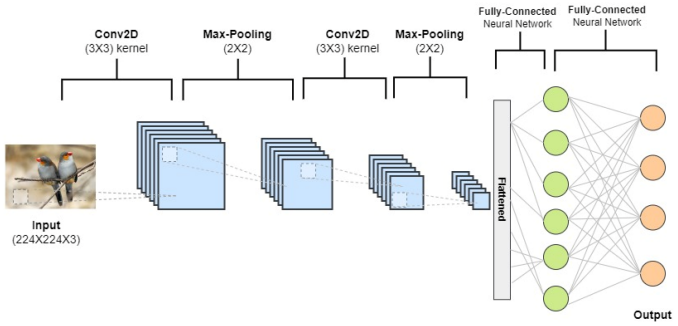


Figure 1: Example of a CNN architecture[1]

CNNs are a successful and established baseline for deep learning in CV which utilises Convolution Layers in conjunction with Pooling and Fully-Connected Layers. Its core operation is the convolution which captures capturing local correlations (homophilly) by applying a filter kernel over input data (like an image) to produce a feature map. Multiple such filters are then used to capture different features like edges, textures, etc.

3.2 Vision Transformer (ViTs)

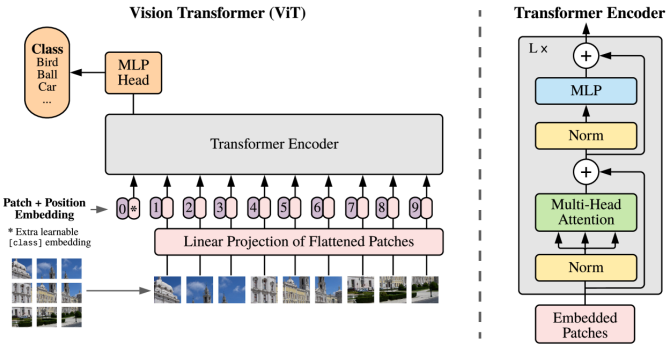


Figure 2: Example of a ViT architecture

Vision Transformers (ViT)[2] are a new paradigm in CV, treating images akin to sequences of tokens in NLP. Its main component is the Transformer Encoder blocks which comprise alternating layers of multi-head self-attention mechanisms and feed-forward neural networks.

Input images are first split into fixed-size patches which are linearly embedded into a flat vector as tokens. A position embedding is then added to each token to introduce the notion

of sequence order. An extra learnable token is introduced to aggregate and accumulate information from other tokens in the sequence over successive encoder layers. This sequence of tokens is then used as input to successive layers of Encoders and the final output of the class token is passed through a linear layer to get the output classifications.

While ViTs have shown promising performance compared to CNNs and are seen as the new paradigm for Computer Vision tasks, they are seen as more computationally demanding and seem, to require larger amounts of training data to achieve comparable performance to CNNs

4 Baseline Model Architecture

4.1 Compact Convolutional Transformer (CCT)

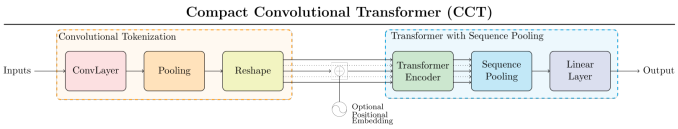


Figure 3: The CCT architecture

Introduced in the 2021 paper "Escaping the Big Data Paradigm with Compact Transformers"[3], the Compact Convolutional Transformers (CCT) architecture aims to merge the strengths of CNNs and ViTs to achieve good performance while being efficient. The models boast competitive results on small datasets such as CIFAR-10 and CIFAR-100 with as few as 0.28M parameters. Their key contribution is the introduction of Sequence Pooling (SeqPool) and the use of Convolutional Tokenisation on small ViTs (ViT-Lite).

4.2 SeqPool

Instead of forwarding an extra learnable class token through the network and feeding it into a classifier, the authors introduce SeqPool, an "attention-based method which pools over the output sequence of tokens". This operation maps the input sequence

$$x_L \in \mathbb{R}^{b\times n\times d} \rightarrow \mathbb{R}^{b\times d}$$

thereby aggregating information across the entire sequence.

This operation consists of mapping the output sequence using the transformation:

$$T : \mathbb{R}^{b\times n\times d} \rightarrow \mathbb{R}^{b\times d}$$

Given:

$$x_L = f(x_0) \in \mathbb{R}^{b\times n\times d}$$

where x_L is the output of an L layer transformer encoder f, b is batch size, n is sequence length, and d is the total embedding dimension. x_L is fed to a linear layer:

$$g(x_L) \in \mathbb{R}^{d\times 1}$$

and softmax activation is applied to the output:

$$x'_L = \text{softmax}(g(x_L)^T) \in \mathbb{R}^{b\times 1\times n}$$

This generates an importance weighting for each input token, which is applied as follows:

$$z = x'_L x_L = \text{softmax}(g(x_L)^T) \times x_L \in \mathbb{R}^{b\times 1\times d} \quad (1)$$

By flattening, the output:

$$z \in \mathbb{R}^{b\times d}$$

is produced.

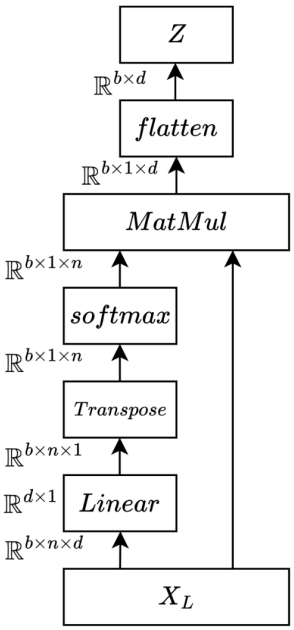


Figure 4: Illustration of SeqPool

This operation hence attends to the entire sequence by assigning importance weights according to the relevance of their positions.

4.3 Convolutional Tokeniser

In the standard ViT architecture, the image is explicitly subdivided into fixed-size non-overlapping patches. Each patch is then flattened and linearly embedded to produce tokens.

For CCT, the authors replace the image patch and embedding with a simple convolutional block. The image is passed through one or more convolutional layers, which transform the spatial dimensions based on the kernel size, stride, and padding. Max pooling is then applied and each spatial dimension of the resultant feature map is flattened and treated as distinct tokens.

Given an image or feature map:

$$x \in \mathbb{R}^{H \times W \times C}$$

The convolutional block operation can be described as:

$$x_0 = \text{MaxPool}(\text{ReLU}(\text{Conv2d}(x)))$$

where the Conv2d operation has d filters.

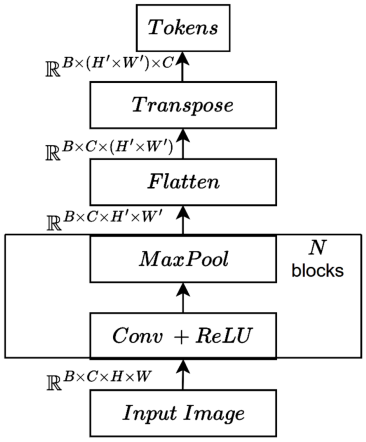


Figure 5: Illustration of the Convolutional Tokeniser

According to the author, this allows the model to maintain local spatial information while producing smaller, richer and more efficient tokens which thereby reduces computation throughout the rest of the model.

4.4 Naming Convention

The combination of SeqPool, the convolutional tokeniser and a series of transformer encoders thereby constitutes Compact Convolutional Transformers. The authors use the convention "CCT-L/PxC", where L is the number of transformer layers, P is the patch/convolution size, and C is the number of convolutional layers.

4.5 Baseline Models Used

In our experiments, we focus on improving the performance of the following CCT model variants:

- CCT-2/3x2 (0.28M Parameters)
- CCT-6/3x1 (3.23M Parameters)
- CCT-7/3x1 (3.76M Parameters)

5 Method and Approach

We tested several approaches towards improving the efficiency of the Compact Convolutional Transformer (CCT) architecture:

5.1 Simple Ensembling

Ensembling techniques aim to combine and aggregate the results of multiple models to obtain better performance than what could be obtained with a single model.

We experimented with this approach by initialising multiple (e.g. 4) small CCT-2/3x2 models and doing the following:

1. Pass the input image through each of these models.
2. Concatenate their outputs.
3. Pass the concatenated outputs through a fully connected aggregation layer.
4. Connect the aggregation layer to the final output layer.

5.2 Dynamic Positional Embedding

CCT originally features a Learnable Positional Embedding of size (1, sequence_length, embedding_dim). In this approach, we pass the output of the positional embedding into the 'DynamicPositionalEmbedding' module which comprises a small multilayer perceptron with 1 input/hidden layer, a ReLU activation, an output layer and a residual connection. The intent is to dynamically adjust the positional embedding based on the input tensor to capture more complex and nuanced position-related patterns.

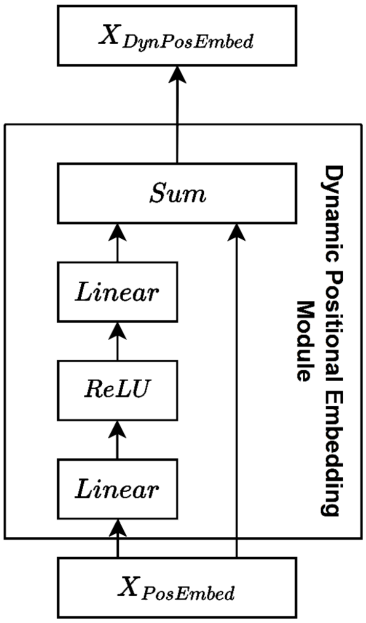


Figure 6: Illustration of the 'Dynamic Positional Embedding' Module

5.3 Trainable Temperature-Scaled Attention

To improve the flexibility of the Attention module, we introduce a learnable temperature parameter into the softmax

function of the attention module class. This allows the model to learn how 'sharp' the attention distribution should be based on knowledge from the dataset during training.

```
class Attention(Module):
    def __init__(self, ...):
        # ... (existing code)
     self.temperature = Parameter(torch.ones(1) * 0.5

    def forward(self, x):
        # ... (existing code)
     attn = (attn / self.temperature).softmax(dim=-1)
        # ... (existing code)
```

5.4 Low-Rank Projection of Linear Layer by a Factor

This technique aims to reduce the number of parameters in a model by factorizing large linear (fully connected) layers into two smaller ones. The idea comes from the observation that neural network weights often have a low-rank structure, meaning that the information can be represented with fewer parameters than initially given.

The feed-forward network in each Encoder block originally consists of two linear transformations with an activation function (GELU) in between. The input dimension (**d_model**) is transformed to **dim_feedforward**and then back to **d_model**.

In our factorised feed-forward network, we introduce three linear transformations instead of the existing two. The input dimension (**d_model**) is first reduced to a smaller dimension (**d_model // dim_reduc_factor**). Then, this reduced representation is expanded to **dim_feedforward**. Finally, it is transformed back to **d_model**.

5.5 Adding SEBlocks to Conv Tokeniser

The Squeeze-and-Excitation Block [4] is an architectural unit designed to improve the representational power of a network by enabling it to perform dynamic channel-wise feature recalibration (PapersWithCode). Given that the Convolutional tokeniser transforms raw image data into a series of tokens, we posit that integrating an SE block with the tokeniser would help recalibrate the initial feature maps before they're fed into the transformer layers. By introducing richer tokens, we could potentially improve convergence and reduce the need for deeper layers without introducing many additional parameters.

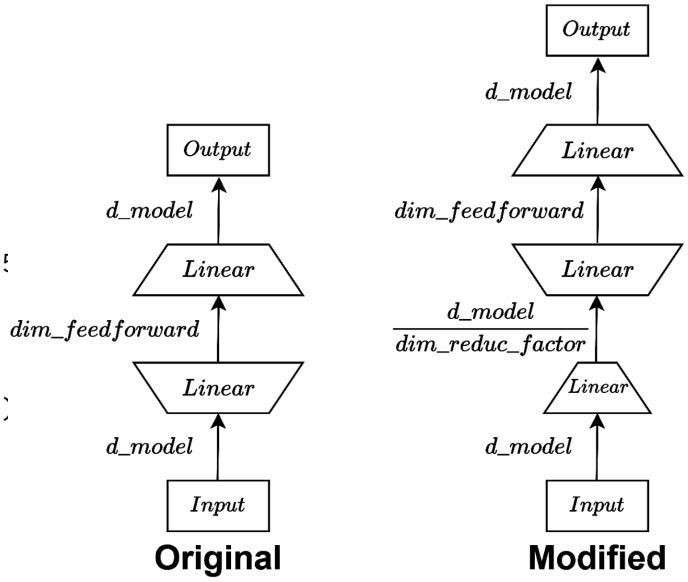


Figure 7: Comparison of the original and modified feedforward networks

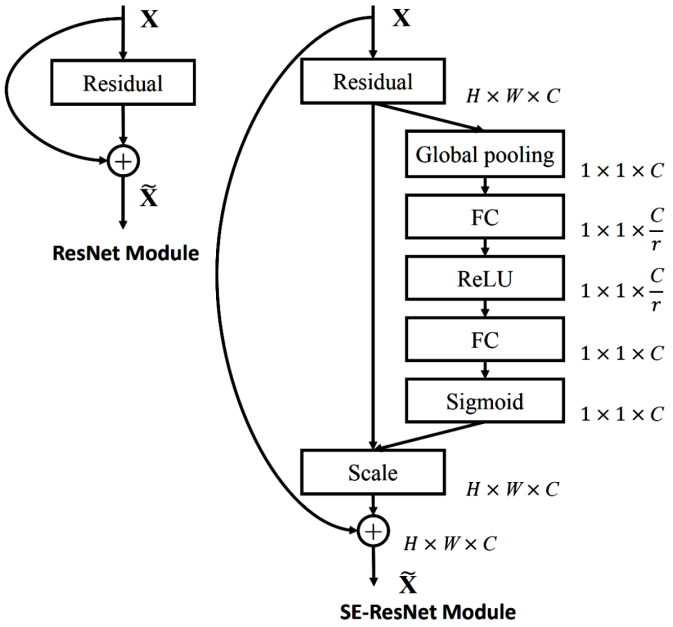


Figure 8: Illustration of the Squeeze and Excitation (SE) Module

6 Experimental Details

6.1 Hardware Used

- Intel i5-13600K
- RTX-3090 24GB VRAM
- 32GB RAM
- 2TB SSD

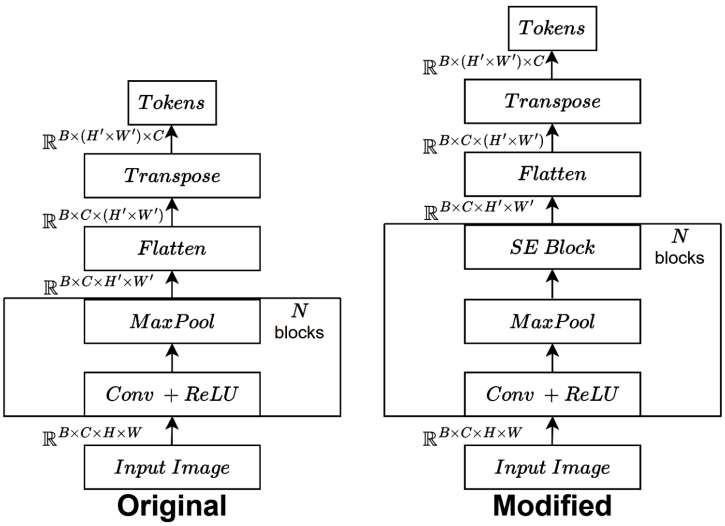


Figure 9: Comparison of the original and modified convolutional tokeniser

6.2 Software Environment

- Python 3.9.13
- WSL2 Ubuntu 22.04.2 LTS
- CUDA Version 12.2
- NVIDIA Driver Version 537.58

6.3 Training

Main Details:

- **Epochs**: 300
- **Batch Size**: 128
- **Learning Rate**: 5.5e-4
- **Warmup Epochs**: 10
  - Training starts with a low LR of 0.00001 before gradually increasing to the main LR over 10 epochs. This helps stabilise the initial training.
- **Optimiser**: AdamW with weight decay of 6e-2

Augmentations:

- **MixUp**: 0.8
  - Encourages better generalization by training on convex combinations of pairs of examples and their labels. This implies that the images and their labels will be a combination of 80% of one image and 20% of another.
- **Cutmix**: 1.0
  - Patches are cut and pasted among training images. The portion of cutmix is set to 1.0, meaning the pasted region will be from another image completely.
- **Random Erasing**: 0.25

Other Details:

- **Loss**: Soft Target Cross Entropy
  - The regular Cross-Entropy loss function expects hard labels. Given that mixup augmentations generate soft labels, it is necessary to use soft (or probabilistic) labels whereby each sample has a probability distribution across all classes representing mixed or uncertain labels.
- **Automatic Mixed Precision (AMP)**: Disabled
  - AMP is not used due to instabilities during training.
- **Label Smoothing**: 0.1
  - A regularisation technique that modifies hard one-hot encoded labels by reducing the confidence of the correct label by the provided value and distributing them across the incorrect labels.
- **Worker Threads**: 8

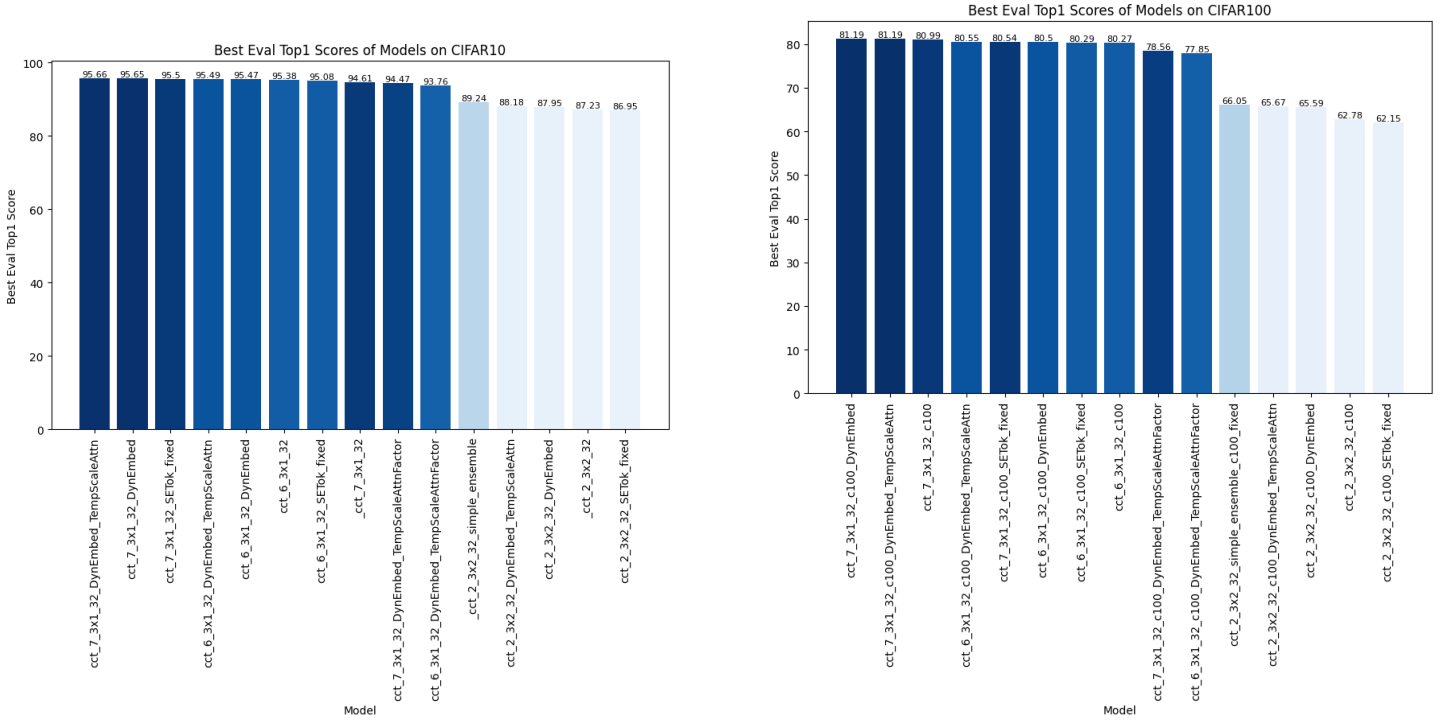
6.4 Evaluation

- **Main Metric**: Top 1 Accuracy
- **Validation Loss**: Cross Entropy

7 Results

Bold represents the best result for the dataset. Brackets represent differences with respect to the baseline models within the same model family (CCT-2, CCT-6, CCT-7)

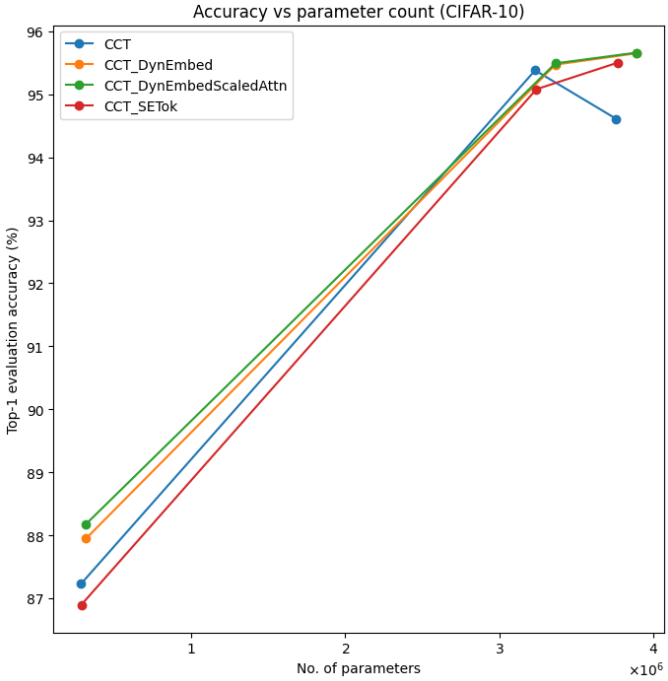
Model	Top1Acc (C10)	Params	MACs	Top1Acc (C100)	Params	MACs
CCT-2/3x2	87.23	283723	0.03324G	62.78	295333	0.03326G
CCT-6/3x1	95.38	3233803	0.81252G	80.27	3256933	0.81254G
CCT-7/3x1	94.61	3760139	0.94674G	80.99	3783269	0.94676G
CCT-2/3x2 4x Ensemble	89.24 (+2.01)	1136942 (+853219;+300.7%)	0.33246G	66.05 (+3.27)	1201472 (+906139; +306.8%)	0.33267G
CCT-2/3x2 DynEmbed	87.95 (+0.72)	316747 (+33024; +11.6%)	0.03534G	65.59 (+2.81)	328357 (+33024; +11.2%)	0.03535G
CCT-6/3x1 DynEmbed	95.47 (+0.09)	3365387 (+131584; +4.07%)	0.84607G	80.5 (+0.23)	3388517 (+131584; +4.04%)	0.84610G
CCT-7/3x1 DynEmbed	95.65 (+1.04)	3891723 (+131584; +3.50%)	0.98029G	**81.19** (+0.20)	3914853 (+131584; +3.48%)	0.98031G
CCT-2/3x2 DynEmbedScaledAttn	88.18 (+0.95)	316749 (+33026; +11.6%)	0.03534G	65.67 (+2.89)	328359 (+33026; +11.2%)	0.03535G
CCT-6/3x1 DynEmbedScaledAttn	95.49 (+0.11)	3365393 (+131590; +4.07%)	0.84607G	80.55 (+0.28)	3388523 (+131590; +4.04%)	0.84610G
CCT-7/3x1 DynEmbedScaledAttn	**95.66** (+1.05)	3891730 (+131591; +3.50%)	0.98029G	**81.19** (+0.20)	3914860 (+131591; +3.48%)	0.98031G
CCT-6/3x1 DynEmbedScaledAttnFactor2	93.76 (-1.62)	3169553 (-64250; -1.99%)	0.79574G	77.85 (-2.42)	3192683 (-64250; -1.97%)	0.79576G
CCT-7/3x1 DynEmbedScaledAttnFactor2	94.47 (-0.14)	3663250 (-96889; -2.58%)	0.92157G	78.56 (-2.43)	3686380 (-96889; -2.56%)	0.92159G
CCT-2/3x2_SETok	86.9 (-0.33)	286487 (+2764; +0.97%)	0.03325G	62.15 (-0.63)	298097 (+2764; +0.94%)	0.03326G
CCT-6/3x1_SETok	95.08 (-0.30)	3242267 (+8464; +0.26%)	0.81253G	80.29 (+0.02)	3265397 (+8464; +0.26%)	0.81255G
CCT-7/3x1_SETok	95.5 (+0.89)	3768603 (+8464; +0.23%)	0.94674G	80.54 (-0.45)	3791733 (+8464; +0.22%)	0.94677G



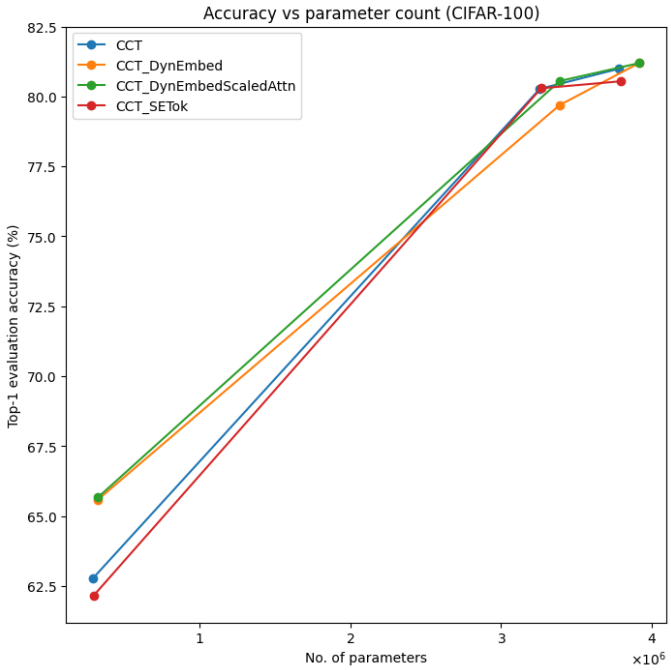
(a) CIFAR10 Models

(b) CIFAR100 Models

Figure 10: Best Evaluation Top1 Accuracy for CIFAR10 and CIFAR100 Models



(a) CIFAR10 Models



(b) CIFAR100 Models

Figure 11: Top1 Evaluation Accuracy against Parameter Count for CIFAR10 and CIFAR100 Models

8 Discussions

8.1 Baseline Models

As expected, the classification accuracy of CCT models generally increases as the parameter count increases. However, CCT-7/3x1 on CIFAR-10 is an exception to this trend, with a lower classification accuracy of 94.61% compared to CCT-6/3x1's 95.38% despite a greater parameter count of 3760139 compared to CCT-6/3x1's 3233803. This suggests that the baseline model may overfit when the model size increases beyond that of CCT-6/3x1, reinforcing the idea that simply adding more parameters to a model doesn't guarantee better performance.

8.2 Simple Ensembling

The performance gain from initialising and ensembling 4 small CCT-2/3x2 models is small relative to the increase in parameters. Our ensemble of 4 such models together only increased the accuracy by 2.01% to 3.27% despite a 300% increase in parameter count.

We believe that the poor performance increase (relative to the increase in model complexity) is due to the lack of diversity among the models. Since the 4 CCT-2/3x2 models have the same architecture, their resulting predictions would also tend to be similar. This highlights the importance of model diversity in ensembling strategies and hints at the need for more sophisticated methods to combine models effectively without unnecessarily inflating parameter counts.

8.3 Dynamic Positional Embedding (DynEmbed)

The addition of DynEmbed led to a small to moderate (+0.09% to 2.81%) increase in accuracy with a corresponding increase (to +3.50% to 11.6%) in parameter count compared to the baseline. The smaller CCT-2/3x2 DynEmbed models have one of the most pronounced increases in accuracy and parameter count, with a +2.81% increase in accuracy and a +11.2% increase in parameter count when evaluated on CIFAR-100.

This increase in accuracy suggests that parameterising the output of positional embedding might be an effective way to increase model performance. However, it is expected that an increase in parameter count leads to an increase in accuracy due to the ability to learn more patterns. Further investigation is needed to determine whether this is an optimal parameterisation of the model to make transformer models more compact.

8.4 Trainable Temperature-Scaled Attention

The introduction of Trainable Temperature-Scaled Attention to the DynEmbed model increases its accuracy by a small amount (+0.00% to +0.23%) with a negligible increase in parameter count (+2-7 additional trainable parameter). Again, the smaller CCT-2/3x2 DynEmbedScaledAttn show the most pronounced improvements.

The modest accuracy increase with essentially no change in parameters suggests that modifications to the attention may be an effective way to make transformer models more compact. However, this performance increase is very minor and could be explained by natural run-to-run variance rather than the efficacy of the modification. Further investigation is hence warranted to determine whether this is an optimal parameterisation of the model to make transformer models more compact.

8.5 Low-Rank Projection of Linear Layer by a Factor

For the CCT-6/3x1 and CCT-7/3x1 models with DynEmbed and Trainable Attention, applying a low-rank projection reduced parameter count by $\tilde{2}$-2.5% while decreasing accuracy by between (0.14-2.43%) compared to the baseline model. This was not implemented for the CCT-2/3x2 model as it increased (rather than decreased) parameter count. The benefits of this are most pronounced in the CCT-7/3x1 DynEmbedScaledAttnFactor2 model which saw a 2.58% decrease in parameter, but only a slight -0.14% decrease in accuracy compared to the baseline.

Our method of down-projection, despite being effective in reducing parameters and MACs, led to tangible performance loss compared to the baseline despite incorporating prior improvements such as DynEmbed and Trainable Attention. This suggests that although this technique could make transformers more compact, may not be the most effective way to do so. There are many other potential ways (architecture and hyperparameters wise) to implement this general idea of down-projecting an intermediate layer which warrants further investigation.

8.6 Adding SEBlocks to Conv Tokeniser

The incorporation of SEBlocks in the convolutional tokeniser resulted in a slight increase in parameter count (between +0.22% to 0.97%) while having a negligible/Indeterminate effect on accuracy; 4 experiments show a minor decrease in accuracy (-0.30% to -0.63%) while 2 experiments show a minor increase (+0.02% to +0.89%).

The general decrease in performance suggests that adding SEBlocks to the convolutional tokeniser is unhelpful as it unnecessarily increases parameter count and MACs without a corresponding increase in performance. One reason for this is that SEBlocks are specifically tailored to augment the output of convolutional layers in deeper convolutional architectures, which play a different role in the overall model architecture

compared to the convolutional layers used in the tokeniser of the CCT model family. More investigations are needed to determine if the SEBlocks can be adapted effectively into this architecture.

8.7 The Value of Negative Results

Although our investigations yielded marginal or negative results, it is still essential to acknowledge the value of publishing negative outcomes. These results, though seemingly unimpressive, are essential in demarcating the limits of effective architectural modifications and guiding future research. Negative findings, especially from well-designed studies, prevent redundant efforts in the field and contribute to a balanced scientific narrative that is often skewed towards positive results.

9 Gradio GUI

For the ease of demonstrating all our trained models, we built a Gradio[3] app that allows users to choose a dataset and one of the 31 trained models for inference on any input images. The display of attention maps is supported for the 4 best performing models: 'CIFAR10 - CCT-6/3x1', 'CIFAR10 - CCT-7/3x1 DynEmbedScaledAttn', 'CIFAR100 - CCT-7/3x1' and 'CIFAR100 - CCT-7/3x1 DynEmbed'.

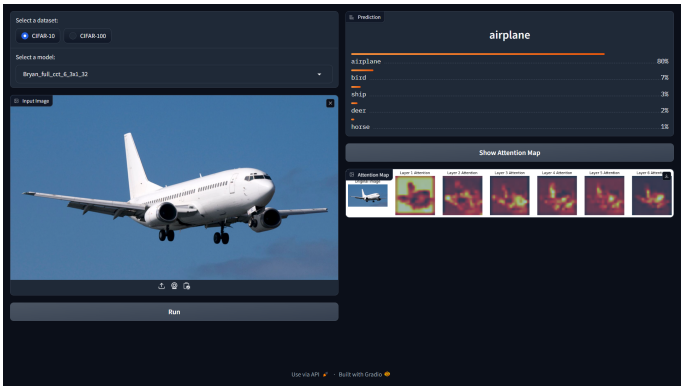


Figure 12: Screenshot of our Gradio App for inference

10 Conclusion

We explored and investigated several techniques in an attempt to improve upon the CCT models. While some paths show promise, particularly with dynamic embeddings and trainable temperature-scaled attention, others like ensembling and SEBlocks integration prove less fruitful. Despite presenting mixed results, this study serves as a stepping stone towards potential ways to improve transformer models without compounding their resource demands.

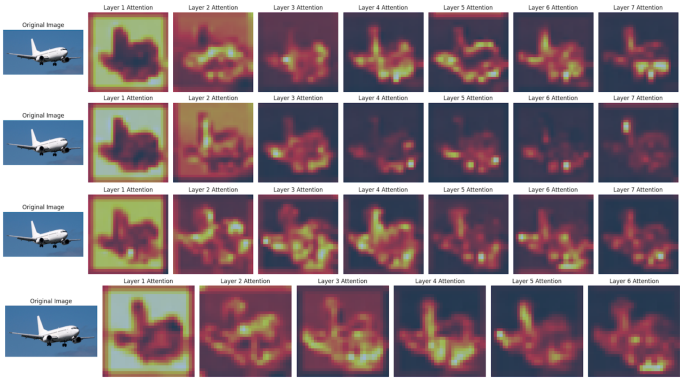


Figure 13: Attention maps for our 'CIFAR100 - CCT-7/3x1 DynEmbed', 'CIFAR10 - CCT-7/3x1 DynEmbedScaledAttn', 'CIFAR100 - CCT-7/3x1' and 'CIFAR10 - CCT-6/3x1' models (from top to bottom)

11 Acknowledgements

We thank Prof. Ngai Mann and Prof. Jun Liu for their guidance and advice throughout the project. We also thank the original authors of the CCT paper for their open-source code.

References

[1] J. Maurício, I. Domingues, and J. Bernardino, "Comparing vision transformers and convolutional neural networks for image classification: A literature review," *Applied Sciences*, vol. 13, no. 9, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/9/5521

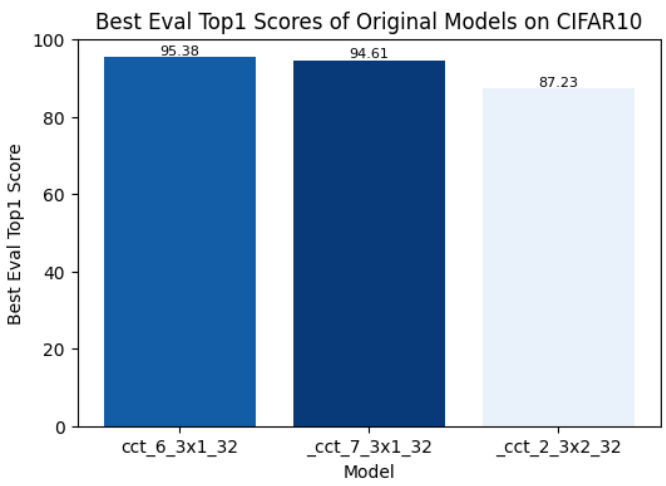
[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[3] A. Abid, A. Abdalla, A. Abid, D. Khan, A. Alfozan, and J. Y. Zou, "Gradio: Hassle-free sharing and testing of ML models in the wild," *CoRR*, vol. abs/1906.02569, 2019. [Online]. Available: http://arxiv.org/abs/1906.02569

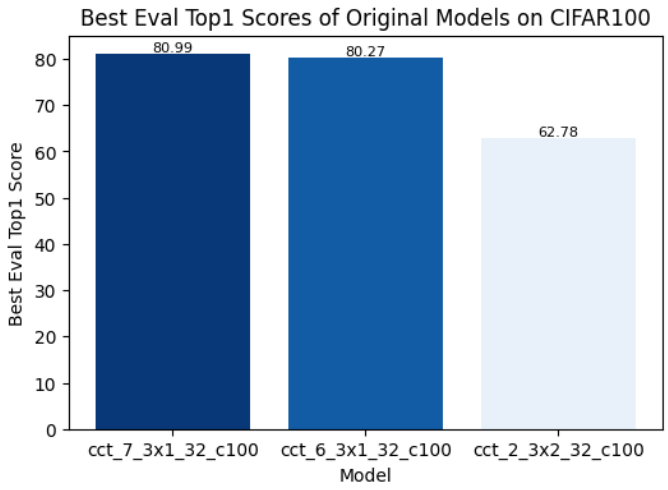
[4] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *CoRR*, vol. abs/1709.01507, 2017. [Online]. Available: http://arxiv.org/abs/1709.01507

Appendix

Model Performance

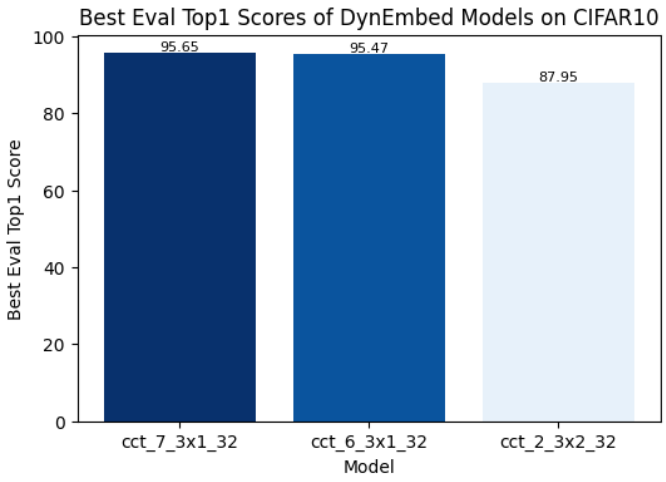


(a) CIFAR10 Models

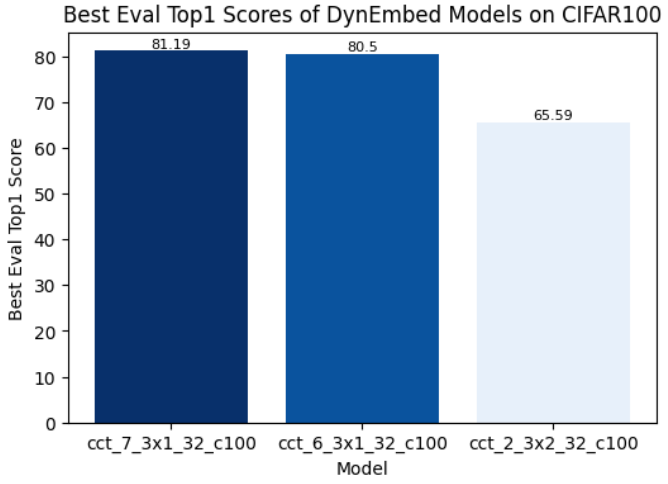


(b) CIFAR100 Models

Figure 14: Top1 Evaluation Accuracy for Baseline Models

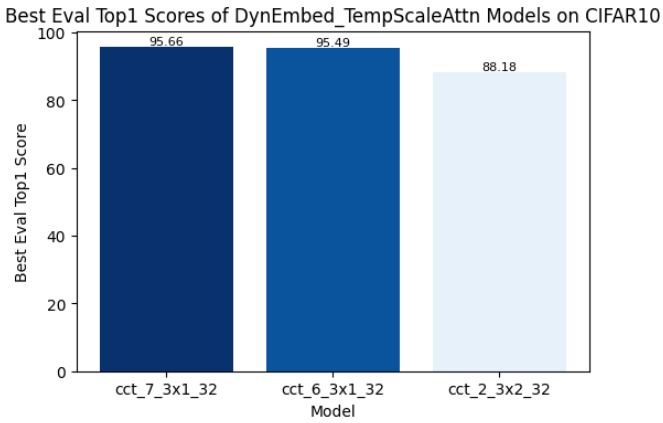


(a) CIFAR10 Models

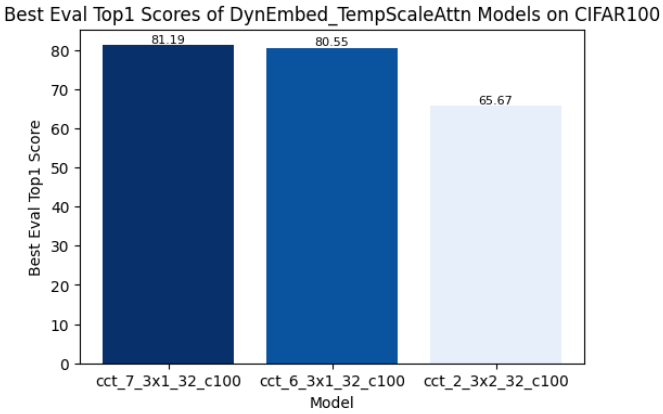


(b) CIFAR100 Models

Figure 15: Top1 Evaluation Accuracy for DynEmbed Models

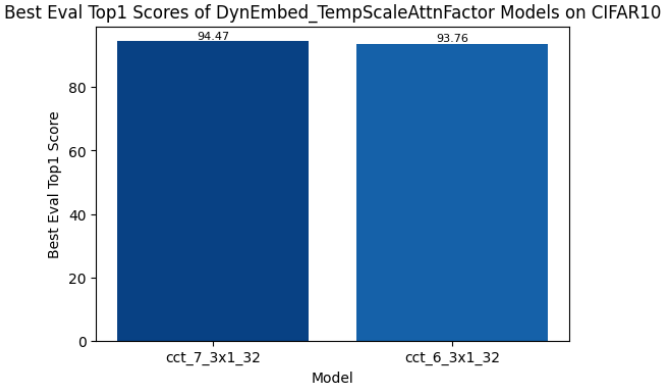


(a) CIFAR10 Models

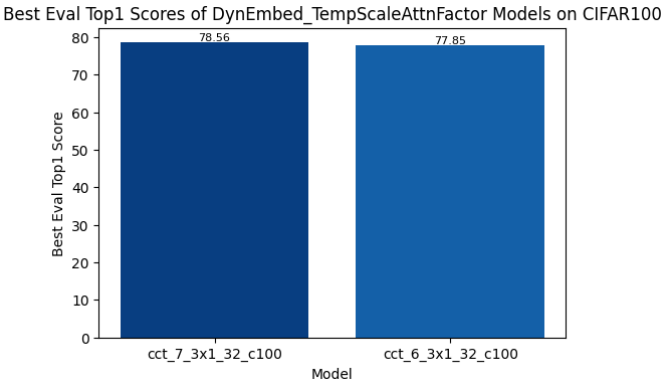


(b) CIFAR100 Models

Figure 16: Top1 Evaluation Accuracy for DynEmbedTempScaleAtten Models

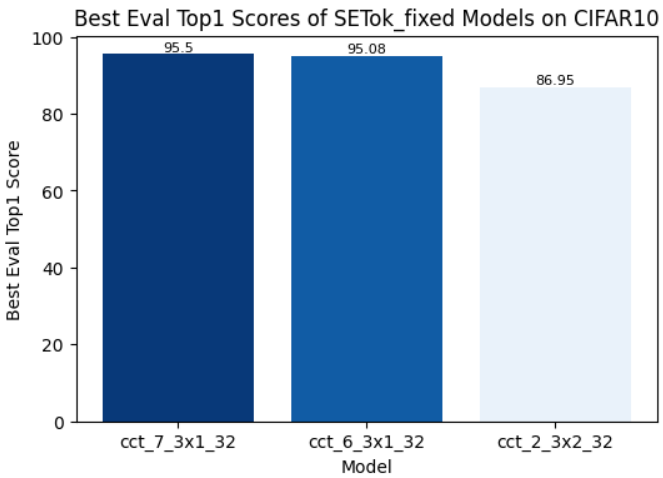


(a) CIFAR10 Models

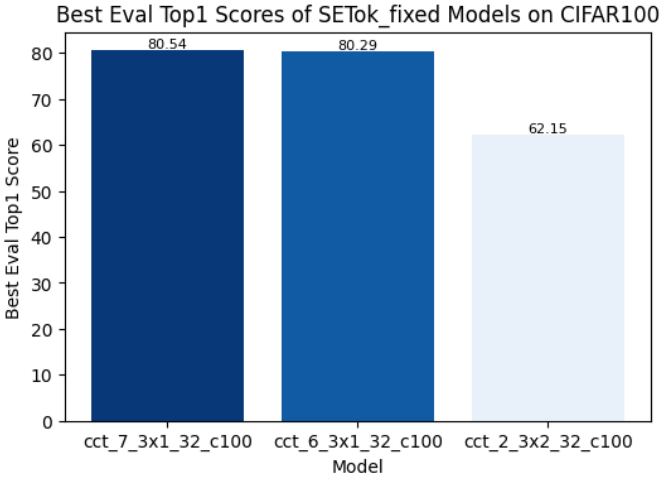


(b) CIFAR100 Models

Figure 17: Top1 Evaluation Accuracy for DynEmbedTempScaleAttenFactor Models



(a) CIFAR10 Models



(b) CIFAR100 Models

Figure 18: Top1 Evaluation Accuracy for SETok Models