Crossing Modalities: Integrating NLP Architectural Innovations into Vision Transformers

Daren Garcia
Dept. of Computer Science
Rice University
Houston, USA
dvg1@rice.edu

Nicole Lewis
Dept. of Computer Science
Rice University
Houston, USA
nl66@rice.edu

Chris Truong
Dept. of Computer Science
Rice University
Houston, USA
ct80@rice.edu

Jake Patock
Dept. of Computer Science
Rice University
Houston, USA
jp157@rice.edu

***Abstract*—Since its proposal in 2017, the Natural Language Processing (NLP) sequence transformer architecture has become a staple in the Machine Learning world [1]. Its brilliant yet simplistic approach of replacing convolutional or recurrent-based architectures with an attention-based architecture expanded the computational ability to analyze more significant inputs and outputs such as images, audio, and video. In this paper, we analyze the efficiency and expressiveness of five proposed architecture advancements of the original visual transformer (ViT): Root Mean Square Layer Normalization (RMS Norm), Gated Linear Unit (GLUs), Expanded Activation, ReZero, and Rotary. Also, four hybrid models using said advancements were benchmarked on a simple image classification task using Imagenette, a smaller 10-class subset of the ImageNet dataset containing approximately 10,000 images, chosen due to computational constraints [21]. The ViTs were evaluated using the macro average precision (MAP) metric in relation to the MAP score of the original transformer architecture. The best model (Hybrid 2 using GLUs, rotary positional encoding, and RMS Norms) resulted in an increase 8.51% MAP over the baseline model from simple architectural modification alone. Given the widespread usage of PyTorch and TensorFlow for NLP tasks, we utilized both frameworks during model building and training.**

Background

Natural Learning Processing Transformers

Note: All code used to produce models can be found in the github repo linked in the footnote of this page [1].

The architecture of the Transformer model has resulted in many advancements in deep learning across various domains [1]. The primary domain where Transformers have been applied most widely is Natural Language Processing (NLP), specifically in tasks such as causal language modeling, machine translation, and sentiment analysis [2], [3]. Most state-of-the-art (SOTA) NLP models are optimized decoder-only variants of the original Transformer model proposed in "Attention is All You Need" [4]–[7]. Leading models in this area include Alibaba's Qwen 2 models, Nvidia's Nemotron models, Meta's Llama series, and Google's Gemma models [4]–[6], [8].

The architectures of the previously mentioned open-source SOTA NLP models differ from the original decoder-only model proposed in "Attention is all you need" [1], [4]–[6], [8].

[1]https://github.com/jakepatock/DL_final_project

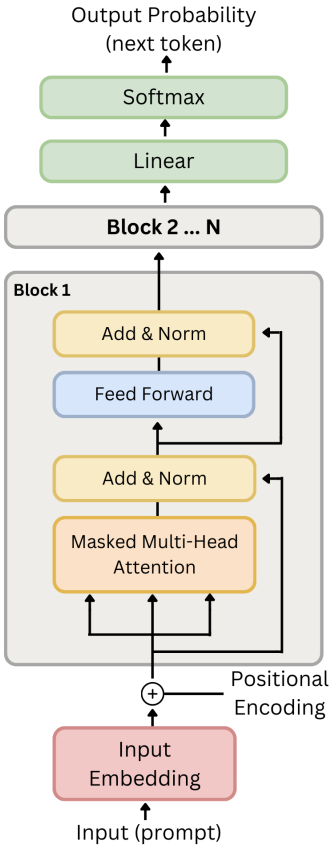

Fig. 1. Diagram of the decoder-only model proposed in "Attention is all you need" [1], [9].

These architectures include iterations on the original decoder-only Transformer, targeting improvements in efficiency and expressiveness. A comparison of a modern decoder-only model Llama 3.1 and the original decoder shows iterations that have been implemented between 2017 and 2024 [1], [4]. The original architecture employs absolute positional encoding, layer normalizations, a post-norm Transformer architecture, and a simple one-hidden-layer feed-forward network [1]. In contrast, Llama 3.1 uses relative positional encoding (rotary positional

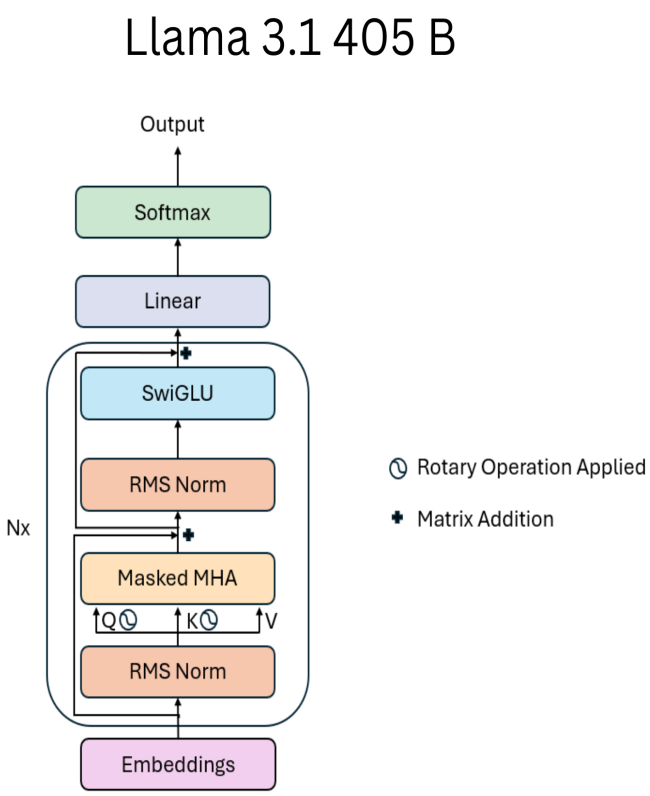

Fig. 2. Diagram of the Llama 3.1 405 B decoder-only model [4]. MHA stands for multi-head attention, defined in "Attention is all you need" [1].

encoding), Root Mean Square Layer Norm (RMSNorm), a pre-norm Transformer architecture, and a Swish Gated Linear Unit (SwiGLU) in the feed-forward network [4]. Modifications such as RMSNorm and pre-norm architecture aim to improve model efficiency, while GLU and rotary encoding enhance model expressiveness [3], [10]–[12]. A visual comparison of Llama 3.1 and the original decoder-only model can be found in Figures 1 and 2.

Original Visual Transformers

While NLP has been the most prominent field for Transformer models, they have also been applied to other deep learning tasks, such as computer vision [3]. In computer vision, the transformer has also produced many SOTA models on benchmark datasets and visual tasks [13]. The Vision Transformer (ViT) model, originally proposed in the paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", is a modification of the encoder-only Transformer variant [14]. Instead of operating on word embeddings, as NLP Transformers do, ViT processes images by dividing them into patches of pixels to derive embeddings. These embeddings are then processed in the standard transformer fashion to perform computer vision tasks such as image classification or object detection [14]. The paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" introduced ViT models of various sizes, incorporating small modifications to the original encoder transformer, such as using a pre-norm variant and applying the GELU activation function to the hidden layer of the feed-forward network [14]. A diagram of this model can be found in Figure 3.

These models were evaluated on several visual tasks, achieving performance comparable to the SOTA CNN-based models of the time [14]. An important milestone was achieved on the image classification benchmark dataset ImageNet [15]. This dataset includes millions of images from about 1,000 diverse classes, making it a challenging benchmark for SOTA models [15]. The largest ViT variant (ViT-H/14) was pretrained on the JFT-300M dataset, then fine-tuned and tested on ImageNet, achieving 88.55% top-1 accuracy on the test partition [14]. This result narrowly surpassed the previous best performance of the EfficientNet-L2 CNN-based model, which had achieved 88.5% [14]. Thus, the ViT established a new SOTA on ImageNet [14].

Recent Visual Transformers on ImageNet

Currently, three of the top four models in the ImageNet image classification task are scaled-up versions of the original ViT architecture, each utilizing different pre-training methods to refine the model [16]–[18]. These models (ranked 4th to 1st) on ImageNet are ViT-e, Model Soup's ViT model, Model Soup's BASIC-L model, and CoCa [16]–[18].

The ViT-e is a larger version of the original ViT, featuring 56 blocks, an embedding dimension of 1792, 16 attention heads, a feed-forward hidden dimension of 15,360 neurons, and approximately 3.926 billion parameters [16]. This model represents a significantly scaled-up version of the original ViT architecture [16]. It was pre-trained on the JFT-3B dataset, Google's proprietary image classification dataset containing about 3 billion images [16]. A noteworthy technique used in its training process was "model souping," where two versions of the same model were trained using differing hyperparameters and combined by averaging their parameters [17]. Specifically, the cool-down phase of ViT-e's pre-training was performed twice: once using inception crop and once without inception crop, creating two distinct ViT-e models with different weights. The averaged weights of these two models produced a "model soup," which improved performance [16]. ViT-e achieved an accuracy of 90.9% on ImageNet [16].

The 3rd and 2nd ranked models used an extended version of the "model souping" technique described above. Both models were pretrained on the JFT-3B dataset, and multiple versions were trained using different hyperparameters (without changing the overall architecture) [17]. The two models with the lowest validation loss were then combined by averaging their parameters [17]. Unlike ViT-e, which only averaged two models, these approaches averaged parameters from multiple top-performing models [17]. The 3rd best model, ViT-G, is another scaled-up version of the original ViT but smaller than ViT-e. It has an embedding dimension of 1664, 48 encoder blocks, a feed-forward hidden size of 8192 neurons,

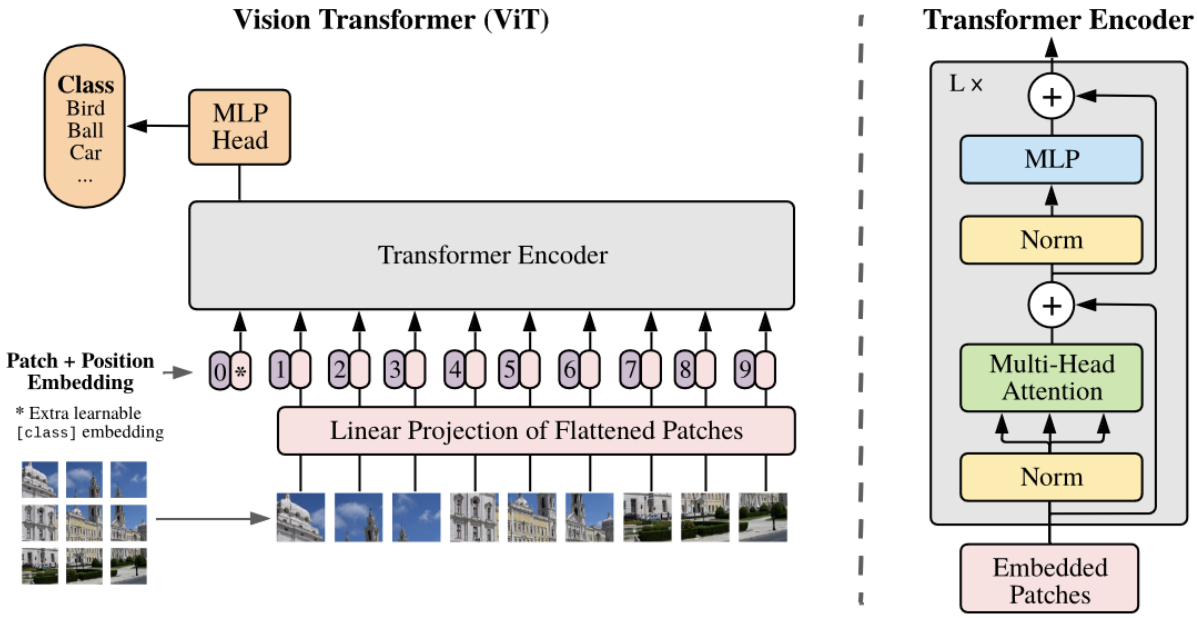

Fig. 3. This is a model overview of the original Visual Transformer (ViT) model that uses a standard transformer proposed in "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" [14]. This is the same architecture used to implement the ViT Base, and ViT Base Premade Models used in this study.

16 attention heads, and 1.843 billion parameters [16]. Using model souping with pre-training on JFT-3B and fine-tuning on ImageNet, ViT-G achieved a test accuracy of 90.94% [17].

The 2nd best model was also part of the "model soup" research but differed in architecture. It was a heavily pre-trained CoAtNet model (a hybrid convolution-transformer architecture) trained on both CLIP and ALIGN tasks, which are collections of multi-modal datasets [17]. Although not a pure ViT, this model achieved an accuracy of 90.98% on ImageNet [17].

The top-performing model, CoCa, is based on the ViT-G architecture but gains performance through innovative pre-training techniques rather than architectural changes [18]. CoCa combines multi-modal pre-training objectives, training a ViT encoder using contrastive loss (similar to CLIP) and generative objectives (as used in SimLVM) [18]. The training aligns the feature map of an image with the feature map of its caption, produced by an NLP transformer decoder. Additionally, a decoder is trained to generate captions for input images [18]. CoCa achieved an accuracy of 91% on ImageNet [18].

It is notable that, except for CoAtNet (a hybrid model), the top-performing models primarily improve upon ViT by scaling up parameters and employing advanced pre-training techniques [16]–[18]. ViT-e scales the original ViT to nearly 4 billion parameters, with pre-training on the JFT-3B dataset and the use of model souping [16]. Similarly, ViT-G and BASIC-L improve performance through model souping of parameters from multiple fine-tuning trials [17]. ViT-G is also a scaled-up version of the original ViT [17]. Finally, CoCa combines ViT-G architecture with multi-modal training techniques, such as image captioning using CLIP-like and generative methods [18]. Interestingly, none of these models significantly modify the original ViT architecture beyond scaling parameter sizes. This highlights an under-researched area in ViTs, as architectural advancements have been abundant in state-of-the-art NLP models today.

Methods: Introducing NLP Modifications into Visual Transformer Models

ViT models are essentially scaled versions of the original architecture proposed in [14]. They incorporate various pre-training strategies and pairings with other encoder/decoder transformer models to train on vast datasets spanning multiple modalities [13], [17], [18]. However, these models do not diverge architecturally from the original ViT in the same way that NLP transformer architectures have evolved. Recent modifications to NLP transformers include substituting layer norms with RMS Norms [10], altering residual connection architectures (e.g., ReZero) [19], expanding activation functions in the feed-forward layer [20], using rotary position embeddings instead of absolute position embeddings [12], and incorporating gated linear units (GLUs) into the feed-forward architecture [11]. Many of these innovations have improved the performance, efficiency, and robustness of NLP transformers [10]–[12], [19], [20]. It is worth investigating whether adapting these architectural innovations to ViT models could enhance their expressiveness or efficiency compared to the original ViT architecture, which remains the foundation for cutting-edge models.

TABLE I
THIS TABLE SHOWS A TABULAR FORM OF WHAT THE TECHNIQUE AND TRANSFORMER ARCHITECTURE THEY TARGETING TO INCREASE THE ROBUSTNESS OF THE MODEL.

Technique	**Architecture**	**Improvement Type**
RMSNorm	Layer Norm	Efficiency
ReZero	Residual Connection	Efficiency
Expanded Activation Functions	Feed-forward Layer	Expressiveness
Rotary Positional Encoding	Positional Encoding	Expressiveness
Gated Linear Units	Feed-forward Layer	Expressiveness

We implemented multiple ViT models, each modified with one of the specified NLP architectural innovations listed above and detailed in Table I. The task was a simple image classification on Imagenette, a subset of the ImageNet dataset. Imagenette is a smaller, 10-class subset containing approximately 10,000 images and was selected due to computational constraints [21]. We combined promising architectures that demonstrated improved performance (both efficiency and expressiveness) to create hybrid ViTs. The original ViT architecture was used as a baseline for comparison. Specifically, the default ViT model used was ViT Base-16 provided by PyTorch, a smaller model chosen due to computational limitations [14]. Additionally, a vanilla implementation in PyTorch was created to compare against the PyTorch prebuilt model. The overall architecture of the base model processes an input image of size 224x224 (RGB). It embeds image patches using a 16x16 convolution kernel with a stride of 16, resulting in "patch embeddings" with a dimension of 768. This produces 196 patch embeddings, as the 224x224 image is divided into a 14x14 grid ($14 \times 14 = 196$). A CLS token with the same embedding dimension as the patches is prepended to the sequence for image classification, resulting in a total sequence of 197 tokens [14]. Absolute position embeddings, derived using the original sine and cosine method, are added to these tokens [1]. The embeddings are then processed by the vanilla ViT encoder blocks, which use a pre-norm transformer architecture. These blocks consist of standard multi-head attention and a feed-forward network with a GELU activation function [14]. The architectural hyperparameters for this model include an embedding dimension of 768, 12 attention heads, 12 stacked encoder blocks, a feed-forward hidden dimension of 3072 ($4 \times$ embedding dimension), and a universal dropout rate of 0.1 applied to both attention and feed-forward layers (at hidden and output stages) [14]. For image classification, the CLS token is extracted after processing and passed through a linear layer to produce a probability distribution over the output classes.

A. Training Hyperparameters and Techniques

All modified models retained architectures as close as possible to the original implementation, apart from the specific modification being tested [14]. Universal training hyperparameters were applied across all models. These included early stopping based on validation loss with a patience of 5 epochs, an initial learning rate of 1e-4, the Adam optimizer (betas of 0.9 and 0.999), and a weight decay of 0. The learning rate scheduler employed a simple reduction on plateau with a patience of 2, a reduction factor of 0.1, and no cooldown. A batch size of 32 images was used for all model variants. Table II shows all hyperparameters used across all model training.

Initial trials in this study produced non-deterministic results, with certain trials showing significant variability in performance even when identical models and hyperparameters were used. This variability likely stemmed from the use of Kaggle's P100 GPU cluster, which caused the virtual machine to behave non-deterministically. Additionally, issues arose with models converging too quickly at higher learning rates, leading to overfitting before the learning rate was reduced. This overfitting effectively negated the benefits of a lower learning rate in enabling the model to approach the local minima necessary for producing generalizable rules for validation and testing. To address these issues, a technique combining a larger patience early stopper with a smaller patience learning rate scheduler was adopted. This method utilized an early stopper that always reset the model to the state with the lowest validation loss achieved and restarted each new epoch from this checkpoint. The learning rate scheduler was configured to wait for two epochs without improvement before reducing the learning rate. This setup allowed the model two opportunities at the current learning rate to achieve a lower validation loss.

TABLE II
THIS TABLE SHOWS THE HYPERPARAMETERS THAT REMAINED CONSTANT ACROSS ALL MODELS.

Hyperparameters	**Value**
Batch Size	32
Image Size	224x224
Patch Size	16x16
Number of Patches	197 (Including CLS)
Embedding Dimension	768
Number of Transformer Layers	12
Attention Heads	12
MLP Hidden Size	3072
Dropout	.1
Optimizer	Adam with Default Parameters
Starting LR	1e-4
Early Stopper Patience	5
LR Scheduler	Reduce on Plateau
LR Scheduler Patience	2
LR Factor	.1

If no improvement was observed, the learning rate was reduced, and the model was given another two attempts at this lower learning rate. If these attempts also failed, the learning rate was reduced again. If a subsequent trial failed, the early stopper was triggered, and training ceased. This combined technique, referred to as "Early Stopping" and "Learning Rate Scheduling with Soft Reset", produced significantly more consistent performance across identical models and training runs. The early stopper and learning rate scheduler patience values of 5 and 2, respectively, were chosen to balance training time with the available computational resources. However, the robustness of this method could likely be improved by increasing the patience values for both the early stopper and the learning rate scheduler.

B. Models

TABLE III
MODEL PARAMETERS AND TRAINING TIME

Model	Parameter Size	Train Time (s)
Base Premade ViT	85,806,346	2652
Base ViT	85,653,514	3867
RMS ViT	85,635,082	3995
GLU ViT	114,001,930	4660
EA ATLU ViT	85,653,526	4343
EA GELU ViT	85,653,526	5220
ReZero ViT	85,616,674	4744
Rotary ViT	85,653,514	4436
Hybrid 1 (Rotary + RMS) ViT	85,635,082	4619
Hybrid 2 (Rotary + RMS + GLU) ViT	113,983,498	4661
Hybrid 3 (Rotary + RMS + EA) ViT	85,635,094	5181
Hybrid 4 (Rotary + RMS + EA + GLU) ViT	113,983,510	5097

1) Base ViT: There are two base ViT models that were tested in this study. The first model (Base PyTorch ViT) was the implementation provided by PyTorch based on the original paper using the CNN embedding technique for each patch [14]. A caveat of this model is that its linear layers use non-dynamically quantizable linear layers, meaning the precision of values in these matrices cannot be dynamically quantized. This model had 85,806,346 parameters [14]. Due to the use of these linear layers, a second base model (Base ViT) was created to ensure consistency across all base models, architectural modifications, and hybrid models. This model, implemented natively in PyTorch, does not use non-dynamically quantizable layers and had a parameter count of 85,653,514.

2) RMS ViT: Although layer normalization is imperative to stabilize any Transformer architecture, it does lead to computational overhead when handling deep neural networks [10]. The Root Mean Square Layer Normalization (RSMNorm) removes the mean statistic for invariance re-centering, which in turn increases efficiency and accuracy [10]. RSMNorm computes the summed inputs into an $\sqrt{n}$-scaled unit sphere, which leads to the expected output distribution without having to hold added computational overhead from the weight distributions and input scaling, as shown in the equations below [10]:

$$\bar{a}_i = \frac{a_i}{\mathrm{RMS}(a)}\gamma, \mathrm{RMS}(a) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}. \quad (1)$$

Where:

- a_i is the i-th value of vector $\bar{a} \in \mathbb{R}^n$.
- $\gamma \in \mathbb{R}^n$ is the parameter used to re-scale the standardized summed inputs.
- n is the dimension of the vector $\bar{a} \in \mathbb{R}^n$.

This formula of the RMS Norm is different from the vanilla layer norm defined below [10]

$$\bar{a}_i = \frac{a_i - E[a_i]}{\mathrm{NF}(a)} * \gamma + \beta, \mathrm{NF}(a) = \sqrt{Var[a]}. \quad (2)$$

Where:

- a_i is the i-th value of vector $\bar{a} \in \mathbb{R}^n$.
- $g \in \mathbb{R}^n$ is the gain parameter used to re-scale the standardized summed inputs.
- $\beta \in \mathbb{R}^n$ is the parameter that shifts the post-scaling normalized vector.
- n is the dimension of the vector $\bar{a} \in \mathbb{R}^n$.

As seen in the formulas, the RMS Norm discards the beta parameter (shift parameter) that shifts the post-scaled outputs and does not calculate the expected value at any point in the formula (instead, the normalizing factor is a root mean square calculation) [10].

RSM Norm reduces the computational cost by roughly 7-64 percent across various NLP implementations due to the significant reduction in computational overhead [10]. In terms of modifications to the Base ViT model, RSM Norm is one of the simplest transformations to introduce due to it only being a formulaic change, as shown in Fig [2] [10]. This is why there is a minimum decrease in parameter size but an increase in the training time of the model in Table III [10].

3) Gated Linear Unit ViT: The Gated Linear Unit (GLU) ViT is a variant of the base ViT architecture designed to enhance model expressiveness by modifying the feed-forward layer. In a standard ViT, the feed-forward layer consists of an input layer, a hidden layer, and an output layer, with the hidden layer employing the Gaussian Error (GELU) activation function [14]. The implementation of a GLU replaces the hidden layer with a gating mechanism that divides the input into two parts: one part undergoes a linear transformation, while the other undergoes a nonlinear transformation [11]. The standard GLU utilizes the sigmoid function and is defined as [11]

$$\mathrm{GLU}(x, W, V, b, c) = (xW + b) \odot \sigma(xV + c)$$

The addition of this gating mechanism increases the parameter count, resulting in a model with 114,001,930 parameters compared to the base ViT's 85,653,514 parameters.

4) Expanded Activation ViT: The expanded activation Vision Transformer (ViT) focuses on the activation function within the feed-forward layer, but instead of implementing a gating mechanism via linear transformers, it modifies the expressiveness when the activation function [20]. Several studies have evaluated the expressiveness of different activation functions by incorporating the ArcTan (ATLU), Gaussian Error Linear Unit (GELU), Sigmoid Linear Unit (SiLU), and Swish (SwiLU) activation functions in the standard transformer architecture [1]. These studies found that GELU performed the best [14], and as a result, these activation functions were implemented into the hidden layer of the ViT model. The self-gating mechanism, which is often associated with the GLU, can also be adapted with these expanded activation functions [20]. In this context, self-gating refers to the model's ability to dynamically control the flow of information within the feed-forward network, where trainable scalars work alongside activation functions like ATLU or GELU to introduce non-linearities and improve feature selection [20]. By replacing the standard activation function with expanded versions ATLU and

GELU (named xATLU and xGELU respectively), the model benefits from enhanced expressiveness, as these functions allow the model to capture more complex patterns and improve learning, particularly in handling high-dimensional data like images [20]. Since this modification occurs at the activation function level, the parameter size increases by one additional parameter per layer. In this case, the total parameter count increased by 12, from 85,653,514 to 85,653,526, corresponding to the number of layers in the ViT [20]. Both xGELU and xATLU demonstrated promising results and were used to create two separate expanded activation function ViTs for this study [20].

5) ReZero ViT: ReZero was a promising architecture modification that targeted the layer norm and residual connection portion of the transformer model [19]. The architecture promised an improvement in model efficiency by decreasing the parameter count (eliminating the need for layer normalization) and producing faster convergence during training [19]. The architecture of ReZero compared to the base ViT is shown here as

$$\text{ViT Base: } x_{i+1} = x_i + \text{MHA}(\text{Norm}(x_i)) \tag{3}$$

$$\text{ReZero: } x_{i+1} = x_i + \alpha_i \text{MHA}(x_i) \tag{4}$$

where MHA is multi-head attention, and alpha is a trainable scalar in ReZero architecture [19]. ReZero eliminates the layer normal and adds the trainable alpha scaler that is initialized at zero and then passes the output from this to the second half of the transformer block (the second layer norm and feed-forward network) [19]. The parameter size of this model was lower due to the elimination of the layer norm, the total parameters for ReZero were 85,616,674.

6) Rotary ViT: The implemented Vision Transformer (ViT) leverages Rotary Position Embeddings (RoPE) to enhance its ability to encode relative positional information within spatial data [12]. Rotary Positional Embedding is an alternative to absolute position embedding. Rotary positional embedding rotates the vectors in the key and query embeddings a set distance depending on the position in the sequence the key and query embedding is in [12]. Comparing the difference in rotation between two different key or query embeddings shows the relative distance the embeddings are from each other. This allows the relative distance of the tokens to be identifiable by the model [12]. This approach replaces traditional absolute positional encoding with sinusoidal harmonic information that cannot be used to compare two embeddings together and derive their relative distance intuitively [12]. This improvement in positional encoding allows the model to capture relative spatial relationships between patches effectively [12].

The process of implementing rotary embedding is applying the rotation on the key and query matrix before it is passed to the key and query weight matrices used in the multi-head attention layer [12]. This way, before the attention map is calculated, the relative embeddings are injected into the attention mechanism [12]. The application of this rotary positional embedding mechanism does increase the number of computations that must be done in the transformer (since it is applied in every block), but it does not introduce an increased number of parameters [12]. Therefore, the parameters of the Rotary ViT implemented here are the same as the base model at 85,653,514 parameters.

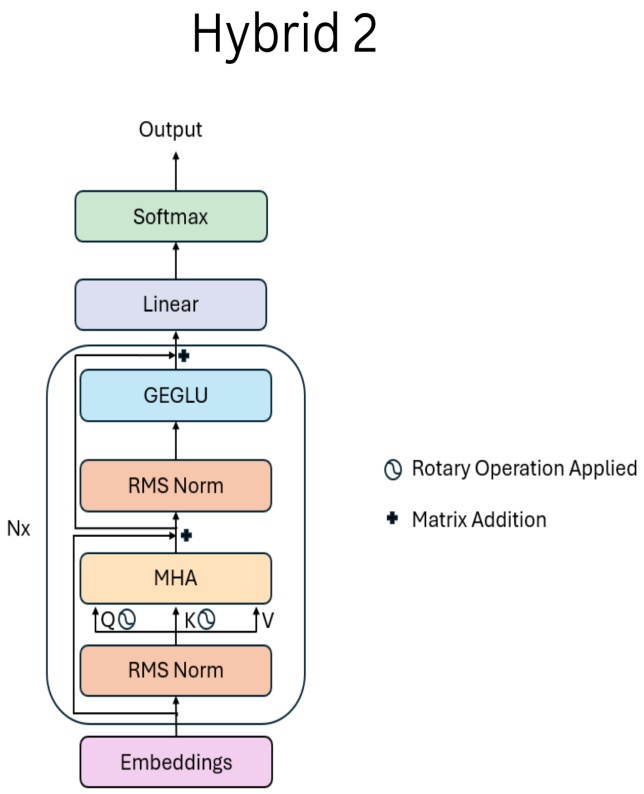

Fig. 4. This is a diagram of the Hybrid 2 model that incorporated RMS Norms, rotary positional encoding, and a gated linear unit feed-forward network with GELU activation. This model achieved the best macro average precision evaluation metric of all models at .7052 (an 8.51 increase of Base ViT proposed in the original ViT paper [14].

7) Hybrid ViTs: The hybrid models constructed in this study were a combination of the most promising architectures. Rotary and RMS Norms produced the most promising results, with both implementations resulting in an increase in all evaluation metrics over the base ViT models. These were the obvious candidates for hybrid model production and were used in every hybrid model constructed. GLU and expanded activation function modification showed slightly lower valuation metrics compared to the base model but were still included in experimentation due to their good results in NLP. Only the GELU expanded activation function was tested in the hybrid model. However, the xGELU performed better than xATLU in the first stage of experiments. Hybrid models produced were Hybrid 1 (Rotary and RMS), Hybrid 2 (Rotary, RMS, and GLU), Hybrid 3 (Rotary, RMS, and GELU expanded activation function), and Hybrid 4 (Rotary, RMS, GELU expanded activation function and GLU). The parameter sizes for hybrid models 1-4 were 85,635,082, 113,983,498, 85,635,094, and 113,983,510.

TABLE IV
THIS TABLE SHOWS THE EVALUATION METRICS OF EACH VIT MODEL.

Model	Validation Accuracy	Test Accuracy	Test Recall	Test Precision
Base Premade ViT	0.6361	0.6030	0.6011	0.6142
Base ViT	0.6705	0.6512	0.6500	0.6513
RMS ViT	0.6725	0.6597	0.6582	0.6589
GLU ViT	0.6720	0.6423	0.6406	0.6442
EA ATLU ViT	0.6595	0.6408	0.6397	0.6403
EA GELU ViT	0.6700	0.6448	0.6435	0.6449
ReZero ViT	0.6033	0.5940	0.5925	0.5923
Rotary ViT	0.7008	0.6866	0.6853	0.6890
Hybrid 1 (Rotary + RMS) ViT	0.7158	0.6881	0.6866	0.6910
Hybrid 2 (Rotary + RMS + GLU) ViT	0.7208	**0.7065**	**0.7052**	**0.7068**
Hybrid 3 (Rotary + RMS + EA) ViT	0.7188	0.6925	0.6913	0.6959
Hybrid 4 (Rotary + RMS + EA + GLU) ViT	**0.7237**	0.7045	0.7034	0.7049

RESULTS

The models produced include implementations in both PyTorch and TensorFlow. However, the result metrics below were derived from PyTorch. Notable differences between the frameworks explored in the discussion section.

C. Model Performances

The primary evaluation metric discussed will be the macro average precision (MAP) metric, which is typical of the standard benchmark for image classification tasks. The results of the baseline models (Base Premade ViT and Base ViT) had test precisions of .6142 and .6513. The Base ViT implemented from scratch performed around 4% better. The GLU ViT's test precision was 0.6442, which was slightly worse but within 1% of the base ViT. Two different expanded activation functions were tested and it was shown that the expanded ATLU function achieved a test precision of 0.6403, and the expanded GELU function achieved 0.6449. These results were again slightly worse than the base model performance but still relativity comparable, being around 1% and .7% off. ReZero ViT showed degraded performance with a test precision of 0.5923. The promising architecture that showed an increase in performance was the RMS and Rotary ViTs. These models achieved test precisions of 0.6588 RMS ViT and 0.6890 rotary ViT. These were both improvements over the original Base ViT. The percent change in MAP performance over Base ViT for all models can be found in Table V.

All the tested hybrid models produced higher MAP scores than any non-hybrid models (including the Base ViT). The MAP for each hybrid were as follows: Hybrid 1 (Rotary + RMS) 0.6910, Hybrid 2 (Rotary + RMS + GLU) 0.7068, Hybrid 3 (Rotary + RMS + EA) 0.6959, and Hybrid 4 (Rotary + RMS + EA + GLU) 0.7049. The best-performing model was the Hybrid 2. The percent change in MAP over the baseline of this model was 8.51%. All MAP for all models can be found in Figure IV, and change in performance percentage can be found in Table V. The architecture of our best-performing model, Hybrid 2, can be found in Figure 4.

D. Model Training / Inference Efficiency

There were multiple efficiency metrics captured. First, the speed of convergence was measured by capturing how many epochs were required in the training phase. The most efficient model for convergence speed was the Base Premade ViT at 14 epochs and all of the models that included GLU at 19 epochs. Common results of this metric were low to mid-20s, with the worst model (ReZero) needing 26 epochs before it reached its final state. Total epochs for all models can be found in Table VI.

Looking at efficiency from the perspective of average task time three metrics were used: Average Epoch Time, Training Step Time (steps/s) and Inference Time (steps/s). The best average epoch time ViT was ReZero (this model has the least total parameters of all models tested). However, the second fastest average epoch was Base ViT, with all modifications increasing the average epoch time over the model. However, the two modifications that increased the average epoch time by the most were the use of GLUs and expanded activation functions. A similar trend was found in the steps metrics.

TABLE V
TEST PRECISION CHANGE OVER BASELINE FOR VARIOUS MODELS

Model	Change in MAP over Baseline
Base Premade	-5.70 %
Base	+0.00 %
RMS	+1.15 %
GLU	-1.10 %
EA ATLU	-1.70 %
EA GELU	-0.99 %
ReZero	-9.07 %
Rotary	+5.79 %
Hybrid 1 (Rotary + RMS)	+6.09 %
Hybrid 2 (Rotary + RMS + GLU)	**+8.51** %
Hybrid 3 (Rotary + RMS + EA)	+6.85 %
Hybrid 4 (Rotary + RMS + EA + GLU)	+8.23 %

The models that showed the most training steps per second were Base ViT 1.5474 steps/s and ReZero ViT 1.6058 steps/s. The worst models for training step efficiency were the GLUs with very large parameter counts (GLU ViT 1.1946 steps/s). The model with the best inference time was the Base ViT again with 2.708 steps/s. Again any model that utilized GLUs performed poorly on the inference time. The GLU ViT had an inference time of 2.0461, which was the lowest of all non-hybrid models. The full results of training steps/s, inference

TABLE VI
THIS TABLE SHOWS THE EFFICIENCY METRICS FOR EACH VIT MODEL.

Model	Train Time (s)	Total Epochs	Training Step Time (steps/s)	Inference Time (steps/s)	Average Epoch Time (s)
Base Premade ViT	**2652**	**14**	1.5474	2.706	189.43
Base ViT	**3867**	21	**1.5912**	**2.708**	184.14
RMS ViT	3995	21	1.5402	2.6119	190.24
GLU ViT	4660	**19**	1.1946	2.0461	245.26
EA ATLU ViT	4343	21	1.4168	2.4217	206.81
EA GELU ViT	5220	25	1.4033	2.4176	208.80
ReZero ViT	4744	26	**1.6058**	2.6778	**182.46**
Rotary ViT	4436	23	1.5192	2.4856	192.87
Hybrid 1 ViT	4619	24	1.5224	2.5263	192.46
Hybrid 2 ViT	4661	**19**	1.1944	1.9991	245.32
Hybrid 3 ViT	5181	24	1.3573	2.2777	215.88
Hybrid 4 ViT	5097	**19**	1.0922	1.8415	268.26

steps/s, and average epoch time can be found in Table VI.

Finally, the total train time for each model was recorded. This metric was dependent on the total epochs and the training step time. ReZero had the fastest average epoch, but the number of epochs ran was much larger, therefore leading to a training time of 4,744 seconds. Therefore, the model that balanced the speed of the convergence metric with the training step time performed the best on the train time metric. The model that balanced these were both of the Base models. They had a lower average of epochs (Base Premade ViT: 14 and Base ViT: 21) while having a high number of training steps per second. A full table of all metrics/results can be found in Table VI.

DISCUSSION

E. Model Performances

The results of this study showed a promising increase in the performance of ViTs using GLUs, RMS Norms, and rotary positional encoding. The rotary position encoding alone produced the largest increase in performance of 5.79 % over all other non-hybrid models. The simple addition of rotary embeddings to the ViT shows an increase in performance without resulting in any additional parameters being added to the model. These results suggest that rotary positional embedding performs a much more robust method for passing positional information about patches to the model. The relationships that are being extracted from images are inherently spatial, and the argument could be made that the brain uses more relative information about parts of an image than absolute information. Since human perspective is dynamic (meaning an object can be viewed from multiple angles, and the perspective can change by simply moving the head), when identifying an object, it is possibly more common for the brain to use the relative position of an object in the image to decipher it. Commonly, the understanding of "the cat is to the right of the dog" is a relative positional statement and a very common way for people to comprehend pictures. This theory is rather speculative and requires interpretation by an expert in human visual abilities.

A surprising result was that the RMS Norm produced a higher performance when the primary purpose behind architects is to increase efficiency. The RMS Norm increased MAP by 1.15% over the baseline. The removal of the beta parameter that shifts the normalized output produced a more expressive model. The reason why this is the case is rather ambiguous. It is possible that this mechanism in the original layer norms commonly models noise with the dataset. This could lead to a decrease in generalization to validation and test sets. The removal of this parameter would prevent the modeling of noise by these parameters (acting as pseudo regularization for the layer norm) and produce a more robust and theoretically more efficient model. If this is the case, it is valid to ask if the trainable Gamma scaling parameter is an effective parameter leading to generalizable rules as well. More comments on this question will be discussed in the ReZero section of this discussion.

Expanded activation functions and GLU modifications on their own produced slightly worse MAP values. The GLU had a decrease in MAP over the baseline of -1.10%. The xATLU and xGELU architectures produced a decrease in -1.70% and -0.99%. These results are very close to the Base ViT model. Running multiple models on differing seeds and calculating a p-value is likely needed to truly discover if there is statistical proof that these methods are indeed lower than the Base ViT MAP performance. This was not done due to time constraints but would be useful to gain more insight into GLU and expanded activation functions. More research would need to be conducted in this area to further examine how both GLU and expanded activation functions would affect the expressiveness of the ViT.

ReZero architecture performed poorly on performance, showing a decrease of -9.07% when compared to the Base ViT. This architecture did not show extreme levels of promise in this study. ReZeros removal of the trainable parameters in the layer norm showed produced much less performance over all. This study shows evidence that the parameters in the Norm layer are truly useful in the expressiveness of the ViT model. However, the conclusions from the RMS Norm ViT lead us to believe that the beta parameter from the layer norm may not be a super robust parameter set. However, when ReZero removed both parameters and the normalization layer overall, its performance was far worse. It may be that much of the expressiveness needed from the layer norm is a result of the

scaling the gamma parameter provides instead of the shift the beta parameter provides. The theoretical explanation of this could be an interesting place for future research. ReZero overall was a poor architecture and highlights how the norms layers in the ViT are both useful for numeric stability and expressiveness of the model.

The results of the hybrid models showed the value of using multiple of these architectures at once produced even better performances. Using RMS Norm and rotary embedding always produced better models over the ViT base (in all hybrid or non-hybrid models). However, when the GLUs and expanded activation function were used in conjunction with RMS Norm and rotary positional encoding, these combinations (Hybrid 2 (Rotary + RMS + GLU), Hybrid 3 (Rotary + RMS + EA), and Hybrid 4 (Rotary + RMS + EA + GLU)) outperformed all the non-hybrid models and Hybrid 1 (Rotary + RMS Norm) as well. This showed that the potential of GLU and expanded activation functions needed pairing with rotary positional encoding and RMS Norms to be fully exhibited. The exact mechanism of how RMS Norms and rotary unlock these other architectures is not fully known and could be a useful vector for more research. It is possible that the robust positional encoding information that is injected by rotary architecture allows for a large amount of new information that these GLUs and expanded activation functions can model. Specifically, the GLU architecture has a very high capacity to fit data but is in danger of modeling noise commonly. The rotary paired with the GLU allows the model to model more universal patterns and cut down on the GLU modeling of noise in the dataset (noise could also possibly be injected by the absolute positional encoding). These results call into question how much of the information injected by absolute positional encoding leads to generalizable results on out-of-distribution data. Rotary positional encoding seemed to be the best modification in this study and allowed other architectural modifications to leverage the information it injects into the embeddings.

The RMS Norm did also increase the MAP evaluation metric but to a lesser degree. It is known that using RMS and rotary embeddings increases performance more when used in conjunction than when used apart. RMS Norm seeded to add a more flat rate of performance increase of around 1%. The current results seem to show that the utilization of the RMS Norm adds a rather flat rate of performance instead, while the rotary technique itself improves performance and aids in other architectures, resulting in improvement in performance as well. The continuation of this study should enumerate more combinations of hybrid models that include hybrids that use rotary without RMS Norms and RMS Norms without rotary. These combinations should illuminate which of these architectures is the main driver that unlocks GLUs and possibly expanded activation functions to push performance.

Finally, the results of expanded activation functions on ViT are rather inconclusive. Both xATLU ViT and xGELU ViT resulted in slightly worse performance compared to the base model. Hybrid 3 (Rotary + RMS + EA) outperformed Hybrid 1 (Rotary + RMS), which showed that expanded activation functions did seem to show some promise for an increase in performance. However, Hybrid 2 (Rotary + RMS + GLU) did outperform this model, showing that GLU does seem to have a higher potential than EA. Another confounding aspect was that Hybrid 4 (Rotary + RMS + EA + GLU) did not outright each Hybrid 2 in terms of increased MAP over baseline and seemed to result in an overfitting of the validation partition. Expanded activations did show some promise in increasing the expressiveness of the ViT, but it was not universal, and GLUs seemed to be the more promising architecture when unlocked by rotary and RMS Norms.

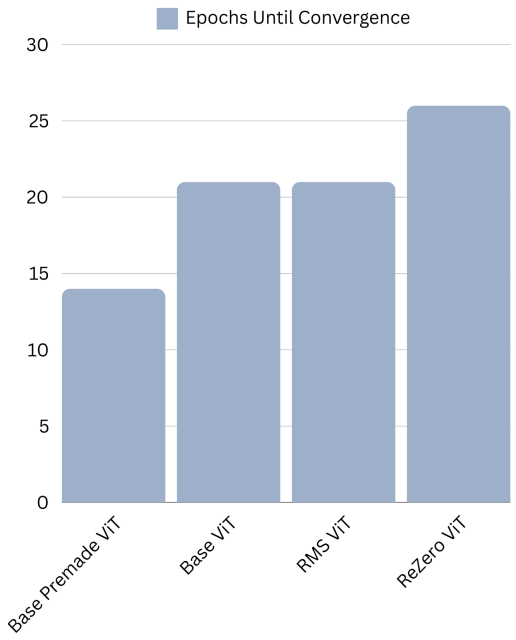

Fig. 5. Bar Chart of the number of epochs it took for the NLP architectures focused on efficiency to converge.

F. Model Training/Inference Efficiency

The NLP architectures that promised increases in efficiency models were not as conclusive. In theory, both architectures, RMS Norm and ReZero, should have increased the efficiency of the transformer by a non-negligible margin. ReZero promised faster convergence times (fewer epochs to convergence) and quicker step times (due to decreasing parameter size) in both training and inference. These two modifications should produce faster training and inference overall. Experimental results of ReZero showed while it did have the highest training steps per second it failed at the promise of increasing convergence. It actually was the model that took the longest to converge at 26 epochs, as shown in Fig. 5. It also did not have the highest inference time step/s either. The Base ViT was the most efficient model at inference time. ReZero suffered from bad convergent to a model where

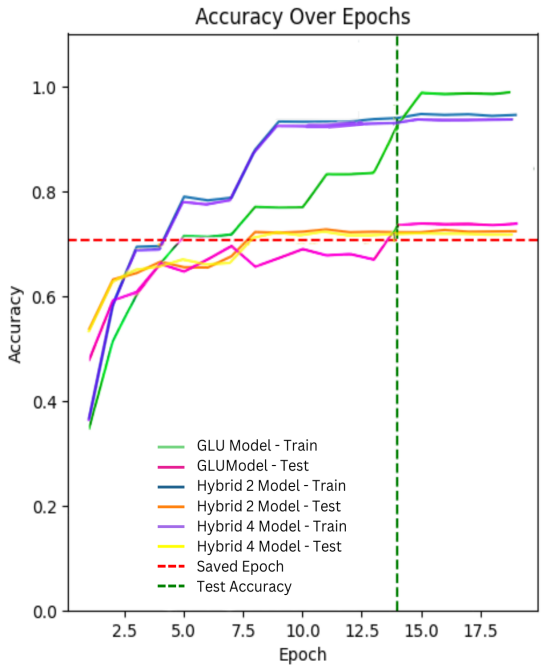

Fig. 6. Accuracy over Epochs for all models using GLUs.

the total training time was high even with the fastest training step. Overall, ReZero did not show promise since the increase in training step efficiency was negated by the decrease in convergence it showed. Efficiency at inference time did not make up for the training efficiency shortcomings either.

RMS Norm was expected to increase the efficiency of the model specifically at the training step time and inference step time metrics. RMS Norm simplifies the calculations done by the original layer norm and also removes the beta parameter, leading to less overall computation overhead. However, this study showed this was not the case in our ViTs. The RMS Norm showed slower training step time and inference time than the base ViT with the layer norms. Both the RMS Norm and Base Norm showed the same number of epochs to convergence. The total train time of the base ViT was also shorter. The base ViT was better or tied to every efficiency evaluation metric used in this study. These results do not make sense as it is well known that RMS does improve performance [10]. One of the explanations for these results could be that the RMS Norm PyTorch implementation that was used in this study was not optimized well and resulted in slower performance than the better-optimized layer norm PyTorch implementation.

An interesting result from this study is that all ViT models that used GLUs in their architecture resulted in a convergence of 19 epochs. While the Base Premade ViT had the lowest epochs to convergence, all GLU ViTs had the second lowest epochs (outperforming all other natively implemented PyTorch models). The GLU architecture reduced the number of epochs required for a ViT model to reach convergence with the specific dataset. Now, the overall efficiency of these models was drastically lower due to the training and inference step times increasing drastically due to the large increase in parameter size that using GLUs brings (Base ViT parameters: 85,653,514, GLU ViT parameters: 114,001,930, an increase of 28,348,416 parameters). GLUs showed an ability to converge faster than all other architectures (even when paired with other architectural modifications). This result does make sense as the GLU adds many more parameters that can be utilized by the model to fit the patterns in the dataset. GLUs were also the worst architecture for decreasing the number of training/inference steps per second. The GLU provides this increased convergence benefit and possible increase in performance but does increase the complexity of the model, leading to much longer training/inference steps and training times overall.

The efficiency of the expanded activation function showed decreased efficiency. The placement of this architect among the architectures was it was more efficient than the GLU but less efficient than RMS and rotary positions encoding. This aligns with the expected results as the complex added by a GLU is much greater than the expanded activation function only. Rotary positional encoding was less efficient than ReZero, RMS, and Base ViTs but more efficient than expanded activation functions and GLUs. Again, the rotary adds fixed computation overhead in each layer, which would be more demanding than simply encoding positional information once before the transformer blocks like what is done in the Base ViT. The rotary, however, was shown to provide a high increase in performance (and allow other architectures to perform better as well) and only had a limited increase in computational complexity, unlike the GLUs, where an increase in some performance cases at a high cost to efficiency overall.

The efficiency results from this study were nuanced. The empirical results form the RMS Norm did not match with the theoretical improvements in efficiency that the architecture has over layer norms. ReZero architectures proved slightly more efficient ViT but not enough to justify the downgrade in performance. The RMS's empirical results did not align with the theoretical increase in efficiency. Rotary embeddings showed a margin loss of efficiency but the performance gains justify it. The expanded activation function showed a loss in efficiency with unpredictable gains in performance. Finally, GLUs provided a decrease in the number of epochs needed to reach convergence but resulted in high costs per train/inference step, resulting in a vastly less efficient model. More research should be done on all models using different GPU hardware than Kaggle's P100 to see if these results hold. Further research into the effects all these architectures have on performance with differing model sizes, training (and pre-training) tasks, and dataset sizes would all aid in understanding the effects on efficiency these techniques have. Also, continuing to test other deep learning frames works like PyTorch vs. TensorFlow would be useful to isolate if some results are a consequence of an imbalanced optimization that might be the case of certain techniques in specific frameworks (like PyTorch

RMS implementation vs. TensorFlow RMS implementation).

G. Framework Comparison

The comparison of results between the TensorFlow and PyTorch implementations of the Vision Transformer (ViT) with Rotary Position Embeddings (RoPE) reveals notable differences and similarities in training behavior, convergence patterns, and final performance [14] [12] [22]. The TensorFlow implementation demonstrates smoother training and validation curves, which reflect more stable and controlled learning dynamics. The training loss decreases steadily and consistently, reaching approximately 1.5, while the validation loss plateaus around 1.73 after epoch 6. Similarly, the accuracy curves exhibit a gradual upward trend, with training accuracy reaching 0.50 and validation accuracy stabilizing at 0.41. These results suggest that the TensorFlow implementation benefits from mechanisms such as learning rate scheduling (ReduceLROnPlateau), early stopping, and regularization (universal dropout rate of 0.1), which together ensure a gradual and stable convergence. The smoothness of the curves can also be attributed to TensorFlow's handling of gradient updates and numerical stability, which appears to reduce abrupt fluctuations during training [22].

In contrast, the PyTorch implementation achieves superior performance, with a final test accuracy of 0.6865 and a significantly lower test loss of 1.10. The precision and recall scores are also notably higher at 0.6890 and 0.6853, respectively, indicating that the PyTorch implementation generalizes better to unseen data. However, the training and validation curves exhibit greater variability. The training loss decreases more rapidly, suggesting more aggressive optimization, while the validation loss demonstrates fluctuations over epochs. Similarly, the accuracy curve rises sharply in the initial epochs but shows intermittent oscillations, particularly in validation accuracy. These sharper transitions indicate that the PyTorch model may have been trained with fewer regularization techniques or different learning rate strategies, leading to faster learning but at the cost of less stable convergence [22].

Both implementations successfully integrate RoPE into the self-attention mechanism, enhancing the ViT's ability to encode relative positional information without additional learnable parameters [12] [14]. This efficiency is reflected in the inference times, which are comparable across frameworks, with TensorFlow reporting an average inference time per batch of 0.1178 seconds and PyTorch achieving slightly faster results with a total inference time of approximately 12.87 seconds. Despite these similarities, the PyTorch implementation demonstrates superior final accuracy and lower test loss, likely due to differences in training dynamics, optimization behavior, or data handling between the two frameworks. However, the smoother curves in TensorFlow highlight its stability during training, which can be advantageous for tasks requiring consistent convergence.

In summary, the TensorFlow implementation excels in producing smoother training dynamics, reflecting stable and gradual learning, while the PyTorch implementation achieves higher accuracy and lower loss but with less stable convergence. These differences emphasize the trade-offs between frameworks: TensorFlow's regularization and learning rate management promote stability, while PyTorch's flexibility and aggressive optimization strategies enable faster convergence and superior final performance [22].

Conclusion

This study analyzed how five different NLP architectural advancements resulted in modified expressiveness and efficiency in visual transformers. The results of this study showed that RMS and Rotary positional embedding improved the expressiveness of the ViT in all cases. Modifications that showed promise in certain scenarios for improved expressiveness were GLUs and expanded activation functions. The ReZero modification was proven to be a degradation to visual transformer architectures when selecting for performance. The best-performing model on metric macro average precision performing an image classification task (trained and tested on Imagenette) was a hybrid model with the architectures of Rotary position encoding, RMS Norms, and GLUs implemented. This model achieved an increase of 8.51% in macro average precision over Base ViT using the vanilla ViT proposed in the original paper [14].

The results for efficiency were less conclusive. ReZero architecture showed promise in increasing the training and inference step time but did not show a faster rate of convergence than other models theorized [19]. GLUs showed an increase in the rate of convergence but a drastic decline in training step and inference step times. Rotary embeddings showed a good trade-off of a small decrease in efficiency but large returns on performance. RMS Norms produced the unexplainable result of decreased efficiency and expanded action function and also decreased efficiency but to a lesser degree than GLUs. Finally, the Base ViT model proposed in the original ViT paper produced the best trade-off of convergence and train/inference step time. Base ViT produced a model with the relatively fastest training time and the best inference speed but lacked performance.

Due to many current SOTA ViT models currently still using the vanilla ViT architecture, advancement in these models' performance and possibly efficiency could be gained from the utilization of these NLP architectures. More research should be done to map the robustness of this modification on larger ViT models trained on more expansiveness datasets and on different visual tasks. These experiments could show the true increase in performance and efficiency these modifications can provide to visual transformers.

Limitations

The limitations of this study are rather dependent on computational and time limitations. Due to computation and time constraints, the only ViT model that was used in this study was ViT Base 16. However, this model is considered very small by current standards. More common models used today are ViT-H (0.632 billion parameters), ViT-G (1.843 billion), or

ViT-e (3.926 billion parameters). By comparison, our largest model trained was our GLU ViT with 0.114 billion parameters. This study fails to study how these architectural modifications would function on a large model with modern parameter sizes. More research should be done on using more modern-sized ViTs and see if the results from this study hold true.

Another limitation of this study is the size of the dataset being used. The Imagenette dataset used in this study is a very small 10-class dataset with about 10,000 images. This dataset is extremely small compared to the actual ImageNet-1k. ImageNet-1k, which contains around 1.43 million images [15]. The limited nature of this dataset prevents insight into how these architectures would perform when trained on larger datasets. Also, more conclusive results on the efficiency of these models would be uncovered when the training dataset is larger and uses more robust hardware than the Kaggle P100 GPU cluster.

One limitation of this study lies in the reliance on A100 GPUs for the TensorFlow implementation of the Vision Transformer (ViT) with Rotary Position Embeddings (RoPE) [23] [12]. While the A100 GPUs provide substantial benefits, such as increased computational power, faster matrix operations, and the ability to accommodate larger batch sizes, these advantages may not be universally accessible in all research settings [23]. The enhanced computational resources likely contributed to smoother gradient updates, improved training efficiency, and reduced training time, which may not reflect performance under more constrained hardware conditions. This raises an important consideration for future work, where further evaluation should be conducted on a wider range of hardware configurations to better understand the impact of computational resources on model convergence, performance stability, and training scalability [23].

This study only had a single visual task to train on as well. The simple image classification task chosen in this study does not encompass the entire breadth of visual tasks. Doing the same benchmarking of Base ViT performance vs hybrid models on other tasks such as object detection, image segmentation, or image ablation studies would be useful research. This other type of task could also improve the image classification tasks via transfer learning. Also, only a set number of hyperparameters were used in this study, defined in Table II. Testing these architectures on more diverse sets of hyperparameters, such as different learning rate schedulers, batch sizes, image resolutions, optimizers, or patience values, would illuminate these effects of the architectures.

Finally, all of these models were trained/tested from initialization on the Imagenette dataset. This does not mimic how modern ViTs are trained and tested. The most common sequence of events is pre-training on an extremely large dataset like Google's proprietary JFT-300M dataset [14]. The model is then fine tuned on the specific task/tasks it will be performing. The models in this paper do not do any pre-training. Therefore, it is unknown how these architectures would perform under the more traditional pre-training and fine-tuning workflow that modern ViTs typically occur in. Research on how using the architecture modification on a model that is pre-training on larger datasets like JFT-3B, ImageNet-1K, or ImageNet-21K and fine-tuning on the specific tasks dataset would be a useful area of study.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of naacL-HLT*, vol. 1, p. 2, Minneapolis, Minnesota, 2019.

[3] Z. Jiang, J. Gu, H. Zhu, and D. Pan, "Pre-rmsnorm and pre-crmsnorm transformers: equivalent and efficient pre-ln transformers," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[4] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[5] B. Adler, N. Agarwal, A. Aithal, D. H. Anh, P. Bhattacharya, A. Brundyn, J. Casper, B. Catanzaro, S. Clay, J. Cohen, *et al.*, "Nemotron-4 340b technical report," *arXiv preprint arXiv:2406.11704*, 2024.

[6] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, *et al.*, "Gemma 2: Improving open language models at a practical size," *arXiv preprint arXiv:2408.00118*, 2024.

[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[8] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, *et al.*, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[9] R. Van Hoorn, "Overview of the (decoder-only) transformer model," Apr 2023.

[10] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[11] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[12] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[13] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022.

[14] D. Alexey, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv: 2010.11929*, 2020.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[16] X. Chen, X. Wang, S. Changpinyo, A. Piergiovanni, P. Padlewski, D. Salz, S. Goodman, A. Grycner, B. Mustafa, L. Beyer, *et al.*, "Pali: A jointly-scaled multilingual language-image model," *arXiv preprint arXiv:2209.06794*, 2022.

[17] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, *et al.*, "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time," in *International conference on machine learning*, pp. 23965–23998, PMLR, 2022.

[18] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "Coca: Contrastive captioners are image-text foundation models," *arXiv preprint arXiv:2205.01917*, 2022.

[19] T. Bachlechner, B. P. Majumder, H. Mao, G. Cottrell, and J. McAuley, "Rezero is all you need: Fast convergence at large depth," in *Uncertainty in Artificial Intelligence*, pp. 1352–1361, PMLR, 2021.

[20] A. H. Huang, "Expanded gating ranges improve activation functions," *arXiv preprint arXiv:2405.20768*, 2024.

[21] J. Howard, "Imagenette: A smaller subset of 10 easily classified classes from imagenet," March 2019.

[22] M. C. Chirodea, O. C. Novac, C. M. Novac, N. Bizon, M. Oproescu, and C. E. Gordan, "Comparison of tensorflow and pytorch in convolutional neural network - based applications," in *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–6, 2021.

[23] W. J. Dally, S. W. Keckler, and D. B. Kirk, "Evolution of the graphics processing unit (gpu)," *IEEE Micro*, vol. 41, no. 6, pp. 42–51, 2021.