1. Introduction

In recent years, the adoption of transformers in natural language processing (NLP) has surged, mainly driven by the success in the development of large language models. This surge in popularity, however, is not confined to the NLP domain as Vision transformers(Vits) have also garnered significant attention due to their impressive use of global attention on various classification tasks **[3]**. With this popularity, many research efforts have worked to enhance the performance of vanilla transformers, especially through the integration of Convolutional Neural Networks (CNNs) and specialized loss functions. While Vits excel at capturing global attention but also requires vast amounts of data[3]. Furthermore, it also misses local fine-grained features, making them more susceptible to noise that corrupts its attention **[1]**. In contrast, CNNs excel at capturing spatial locality and translation invariance, allowing them to require significantly less data to train effectively. However, CNNs suffer from a lack of global attention, which restricts their abilities to capture long range dependencies present in the data **[1]**. Given their complementary strengths and weaknesses, a substantial body of research has been focused on integrating CNNs with Vits to leverage both their advantages. Despite their impressive results, these models often distort the global attention characteristic of transformers to introduce spatial locality inherent in CNNs. [2] additionally, a growing body of work has been focused less on architecture and more on the loss functions used to train the models. One such method, ArcFace loss, incorporates margins to more effectively reduce intra-class distance and increase inter-class distance. In this paper, I propose a novel approach to addressing the challenge of combining Vits with CNNs by utilizing CNNs to generate attention weights on individual patches and leveraging ArcFace loss to better guide the attention of the model. I applied the new model, FaceRec, to the extensively studied domain of facial recognition, utilizing it to benchmark measure against established standards.

2. Related Work

Recent research into combining convolutional architectures with Vision Transformers has primarily focused on Architectural Integration approaches, specifically parallel and sequential integration [1]. In parallel integration, CNNs and ViTs operate in parallel, each presenting a feature vector that is then combined together [1]. For instance, models like the Conformer leverage CNNs to capture local spatial features and ViTs to extract broader global information, allowing data to flow between the two models [1]. However, the merging of these features also leads to interpretability issues and as such also begs the question of whether the spatial locality of the CNN, which is easily learned, dominates the global attention of the ViT, which takes longer to learn. In contrast, sequential Integration uses a different approach, as it passes Convolutional embeddings straight into the Vision transformer, utilizing the spatial locality gained within the Convolutions and applying global attention on them. An example of one such model is Compact Convolutional Transformers (CCT), which has shown boosted performance in top 1 accuracy benchmarks [2]. However, such models lose global attention, as the Vits apply attention not on the global patches but on the spatial locality of the filters [2].Overall, Architectural integration provide an innovative approach in combining the spatial attention of CNNs with the global attention of ViTs, but suffer uninterpretable architecture, loss of important attention information, and the added training and computational complexity of training two separate models. However, these approaches provide a foundational framework for integrating CNNs with ViT Architecture, specifically by leveraging CNNs to add information to Vision Transformers. Beyond architectural integration, another field of emerging interest is the combination of attention to emphasize and suppress specific CNN outputs. This weighted attention allows the model to emphasize the most critical channels while suppressing less important ones. For instance, models like Squeeze-and Excitation Networks (SE-Net) utilize an innovative "squeeze and excitation" method to learn interdependencies between the channels of its features [5]. Alternatively, as the popularity of Multi head attention layers has risen in recent years, models such as BoTnet integrate MHSA directly into CNNs, like CCT models [6]. These models essentially leverage the global attention provided by self-attention models to better understand the interplay between the different convolutional features. While this attention-based approach has shown significant performance in understanding these interdependencies, it is designed primarily for CNN architecture performance and can significantly increase the models' sensitivity to noise. [1] However, research into this area does raise a critical question, specifically can CNN attention mechanisms be incorporated into vision transformers without distorting model's capabilities?

With attention mechanisms in focus, we examine Key point Relative Positional Encodings, an innovative approach within the facial recognition domain. KP-RPE leverages the distances between key points, such as the eyes, nose, and mouth, to guide the attention mechanisms of vision transformers [7]. By computing patch distances and subtracting them from key point coordinates, we can create a Distance matrix that when added to the transformer's attention matrix weights the attention toward critical key points [7].This method has proved remarkably effective, but it has two significant limitations: Its applicability is largely restricted to facial recognition tasks and its reliance on predefined key points making it less adaptable to other classification domains where imperceptible additional features could enhance performance. To address this, convolutional filters emerge as a compelling solution. CNNs excel at capturing spatial locality and as such can capture nuanced features that can be far more significant than key point distances. For instance, unique texture or skin tone patterns identified by CNNs could be more significant indicators of identity. Furthermore, CNN filters wouldn't be just constrained to one single domain, but rather spread across the entire image classification space. However, this increases the model complexity and jeopardizes performance due to noise, as research from previous attempts show. My solution to this is utilizing earlier research into Arc Face Loss to help guide the training of the model. My hypothesis is that forcing stricter margins in between classes will rectify these challenges. I chose ArcFace loss specifically due to its relative simplicity compared to current cosine losses and its better performance [4].

3. Method

For the method, please reference the architecture in figure 1 and figure 2 in the Figures section. I propose a novel approach where the individual patches, in parallel to the transformer embedding layer, are passed through convolutional blocks to create spatial embedding representations of each patch. These representations then are formed into an attention matrix which is added to the global attention matrix created by the vision transformer in the MHSA layer. Finally, I utilize classification research to add a sequential layer for more in-depth classification features and ArcFace loss for stricter class centers. The architecture begins with the input image X, a C x H x W matrix. This matrix is divided into N x P x P x C pixel patches, which we will denote as X'. These patches are reshaped to size N x P * P x C, before being passed through an embedding layer which multiplies a linear projection weights matrix W to create embeddings of N x d. Here d is the embedding dimension of the transformer. The resulting projection is then combined with a precomputed sinusoidal position embedding. Here N is the number of patches, C the number of channels in image.

$$X_v = X'_{(N \times p2 \times C)} \times W_{((C \times p2) \times d)} + Positional\ Encodings_{(N \times d)}$$

$$transformer_{output} = transformerBlock(X_v)$$

In parallel X' is fed through a custom Convolutional Embedding Layer. This layer is composed of three convolutional blocks, where each block is made up of a Convolutional layer, a Relu Layer, and a Max pooling layer. The output of which is then input into a custom multi attention head that returns just the attention matrix. The result is an attention score between the spatial embeddings for each patch. This design is inspired by the research of combining CNNs with attention mechanisms, where the interdependencies between filters can prove to be important attention weight to the vision transformer. The addition of the CNN attention matrix with the global attention matrix takes inspiration from the KP-RPE addition of distance attention to global attention. My hypothesis is that convolution filter embeddings can contribute meaningful spatial attention that can better guide the transformer to learn the interdependencies between patches, without corrupting the global attention of the vision transformer. Here dc is the embedding size of the convolutional embeddings and Wq, Wk are weight matrices found in MHSA. A is the number of attention heads.

$$Conv_{Embedding(N\ x\ dc)} = \bigcup_{i}^{N} ConvBlock(X'_{(N\ x\ P\ x\ P\ \ x\ C)})$$

$$K_{(A\ x\ N\ x\ dc)} = reshape(Conv_{Embedding} \times W_{k(\ dc\ x\ dc)})$$

$$Q_{(A\ x\ N\ x\ dc)} = reshape(Conv_{Embedding} \times W_{q(\ dc\ x\ dc)})$$

$$Conv_{Attention(\ A\ x\ N\ x\ N)} = \frac{(QK^T)}{\sqrt{d_c}}$$

Then in the MHSA layer of the transformer we add this matrix to the global attention matrix. Here V is the values matrix of the vision MHSA layer.

$$softmax(Vision_{Attention} + \ Conv_{Attention})\ x\ V_{(A\ x\ N\ x\ N)})$$

After the embeddings have been passed through the transformer, they contain important information on each patch. To leverage all this information, we take inspiration from the Sequential layer implemented by CVT [2]. This sequential layer is a Multi-layer perceptron which projects each patch's embeddings into a scalar value, essentially a weight for each patch. This weight is then multiplied by the transformer output to produce the feature vector. This is done so that each patch can contribute to the final classification, rather than one gaussian distributed classification token. This produces the classification feature vector.

$$softmax(MLP(transformer_{output}))_{1\ x\ N}\ x\ transformer_{outputN\ x\ d}$$

Finally, these logits are then passed to an ArcFace Layer, mentioned in this paper [4]. Essentially the main idea is that before passing logits to a Cross Entropy Loss, we project each feature vector and weight center onto a hypersphere. We then scale down the hypersphere using normalization so that the distance between the feature vector and the center weights is solely based on theta: the angle between them. By only increasing the theta of the respective correct label by a marginal distance m, it will make the boundaries between classes more distinct. With these distinct boundaries, the model must now more strictly classify that feature vector for the class, essentially training a more discriminative model. For a more visual example, see figure 2. To achieve this, we utilize the following cosine similarity equation.

$$W^T feature_{vector} = ||W|| * ||feature_{vector}|| * \cos(\theta)$$

Using algebra, we can reformat the equation as

$$arccos(\frac{vector}{||W|| * ||feature_{vector}||}) = W_T feature \qquad \theta_{(\ 1\ x\ classes)}$$

This theta vector is now the theta angle for each class in relation to the feature vector. Now we take the class label associated with the specific feature vector and create a one hot encoding for that specific class theta. After that we just add m to that specific class theta and reapply cos.

$$s * \cos(\ [\theta_1, \theta_2, \theta_3 \dots \theta_i \dots \theta_{classes}] + [0,0,0 \dots m \dots 0\ ])$$

We then multiply this theta feature vector by a constant scalar s to scale the unit hypersphere. ArcFace produces two logits: the classification logits without the margin m used for the purpose of classification, and the loss logits with the margin m. These loss logits would then be passed onto a Cross Entropy Loss and used in the backpropagation.

4. Experiment

For specific hyperparameters, reference figure 3. I trained the FaceRec model on the CelebA Images dataset of size (256 x 256) with 7,342 training samples, 1839 testing samples, and 306 classes. I used a Cross Entropy loss function, an Adam optimizer and a Cosine Annealing Warm restart learning rate schedule. I decided to evaluate the model on top 1 accuracy and ArcFace loss and to compare it to the performance of three distinct models: The vanilla Transformer, the KP-RPE model, and the CCT model. My first approach, seen in figure 4, proved to be very useful as I saw the model's training loss decrease from around 37.0 loss to near 0.12 loss. The testing loss started at 7.0, but stagnated, to 5.21. I hypothesized that the model was overfitting to the training data and introduced a weight decay of 0.1 to the Adam optimizer, which would penalize the model for larger. I also noticed that while the testing loss decreased it was far less than I expected. One possible hypothesis I had was that the margin of 0.5 was too large for the initial model and as such it was struggling to even find the optimal weight centers. As such I decreased the margin to 0.35 to reduce the strictness of the loss, increased the initial CNN embedding dimension from 256 to 512, and increased L from 8 to 10. I also changed T_init for the LR scheduler to 500, for a faster restart. On approach 2, as figure 5 shows, the model did not fall as much into overfitting, but it still performed very poorly on the testing. From my analysis ,the test dataset had low loss and, as in figure 6 shows, a low top 1 accuracy of 20.36. As such, with such a low testing accuracy, high training loss, and low testing loss, it meant that the model was learning the difference between identifying correct and incorrect classes, but it couldn't make leaps into accurate classification for unseen data.

5. Conclusion

In conclusion, this experiment, while not meeting benchmark performance, is an innovative approach to introducing CNN features to Vision Transformers. Furthermore, the use of Arc Face loss introduces a strict guide in training the model. However, due to the computational resources available, not enough data could be used to train the model, so next steps would be to efficiently train the model on more data, more features, and regularization like dropout. Overall, this was interesting initial approach into combining CNNs, ViTs, and ArcFace Loss.

6. References

[1] Haruna Yunusa, Shiyin Qin, Abdulrahman Hamman Adama Chukkol**2** Abdulganiyu Abdu Yusuf, Isah Bello, Adamu Lawan "Exploring the Synergies of Hybrid CNNs and ViTs Architectures for Computer Vision: A survey" arXiv (9 - 12)

[2] Ali Hassani1, Steven Walton1, Nikhil Shah1, Abulikemu Abuduweili1 , Jiachen Li2,1, Humphrey Shi1, "Escaping the Big Data Paradigm with Compact Transformers" arXiv

[3] Alexey Dosovitskiy, LucasBeyer,Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby,"An Image is worth 16x16 words: Transformers for Image Recognition at Scale.

[4] Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou "ArcFace: Additive Angular Margin Loss for Deep Face Recognition" arXiv

[5] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu "Squeezeand-Excitation Networks" arXiv

[6] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, Ashish Vaswani "Bottleneck Transformers for Visual Recognition" arXiv

[7] Minchul Kim, Yiyang Su, Feng Liu, Anil Jain, Xiaoming Liu "KeyPoint Relative Position Encoding for Face Recognition" arXiv

7. Figures

Figure 1: FaceRec Architecture

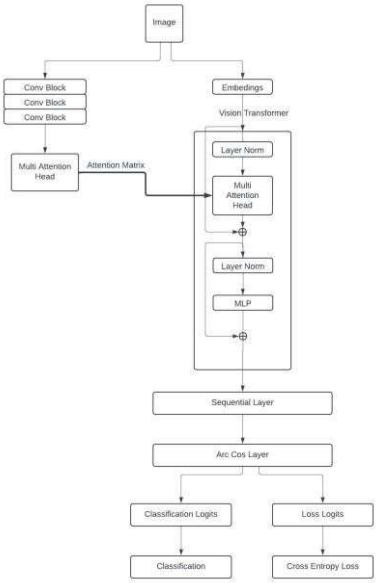

Figure 2: ArcFace Loss Visualization Figure 6: Accuracy Comparison

Model	Top 1 Accuracy
FaceRec	**20.36**
Vision Transformer	88.55 [3]
CCT	98.00 [2]
KP-RPE	94.28 [7]

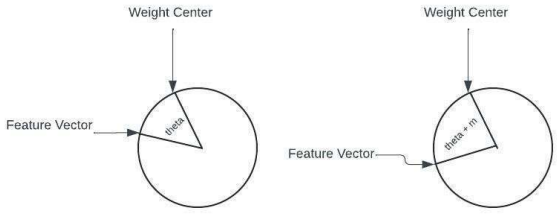

Figure 3: Hyperparameters

Hyperparameters	**Values**
Attention Heads	16
Transformer Embedding	1024
CNN Embedding, {filter sizes}	{ 256 , 512 }, { 2x2, 3x3 }
Learning Rate, {T_init, T_mult}	.001, {{1000, 500 }, 1}
Batch Size	128
Hidden MLP scale	3
Epochs	{ 12, 15 }
Patch Size,	32 x 32
L	{ 8 , 10 }
Arc Face S , margin m	64, { 0.5}
Hardware, time	T4 – GPU, {4 hours, 4 hours}

Figure 4: 1st Attempt

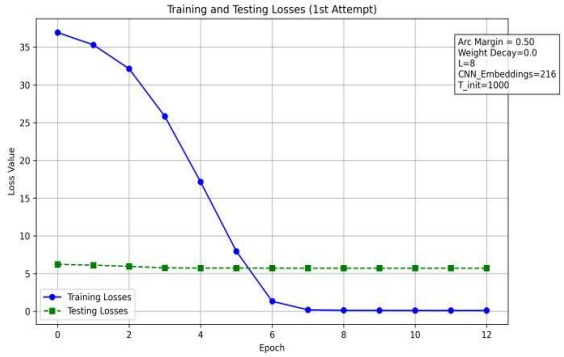

Figure 5: 2nd Attempt