Visual Transformer

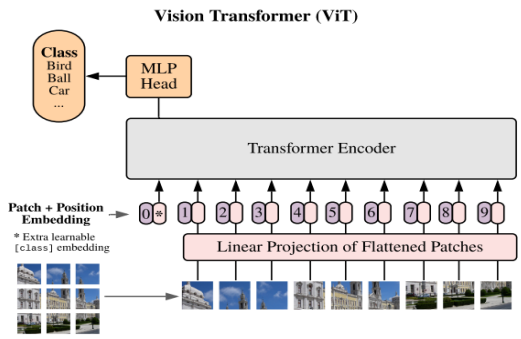


Figure of VIT Architecture

Visual Transformer from google uses transformer architecture that is based on text-based tasks. Overall approach of this model starts with taking original image and dividing it into patches. These patches can be *6x6 or 16x16* and flatten out into sequences and pass these patches into transformer layer. This is similar to transformers using position embedding where these embeddings indicate the location of the left or right order of text sequences. In the case of *VIT,* embeddings allow transformer layers to indicate where each patch were sourced from the original image. Finally, these embeddings are added to the original patches and passed into transformer encoder/decoder where it can be any kind of architecture that is compatible with patch implemented input representation.

Hyperparameter of VIT

Layer	*Variable*	*Value*
Inputs	*Input_shape*	*(128,128,3)*
inception	*Image_size*	*128*
	Patch_size	*10*
	Num_patches	*(Image_size//patch_size) ** 2)*
Transformer	*Projection_dim*	*64*
	Num_heads	*4*
	Transformer_units	*[Projection_dim * 2, projection_dim]*
	Transformer_layers	*4*
Multi-layer perceptron	*Mlp_head_units*	*[2048, 1024]*

Table of VIT Hyper-Parameters

Visual Transformer Is a complex deep layer model, and hyperparameter of the transformer layer significantly affects the end results thus, it is essential to talk about the hyper-parameters used in this model. Learning rate/weight-decay *(0.001, 0.0001)* is used in *Adam-optimizer*. *Patch size of 6* is going to be size of the patches where original 128x128 image is transformed into grid size of 6x6 crop. Then next parameter is to store *number of patches* in a variable where transformer layer is going to be using when defining the position embedding. Size of hidden dimension feature vectors *(projection dim =64)* is to project these patches into these 64 feature vectors then concatenate together as input into the first

transformer encoder layer. Next hyper parameter is number of attention heads *(num_heads = 4).* In multi-head self-attention, this demonstrates having four different parameterizations of the query key and value matrices. Four separate transformations then aggregate the outputs of these four separate self-attention parameterizations from the previous layer. *Transformer unit lists* allows to override the multi-layer perceptron layer (MLP layer) to add skip connection that is like Resnet. In transformer block, it has skip connection from the output of self-attention then it goes through a forward layer that is going to compress the dimensionality then it skips ahead with previous output from the previous dense layer. Finally, transformer blocks (*transformer_layers = 5)* which demonstrates stacking top of the transformer encoder block that repeats it five times then there is hidden dimension of multi-layer perceptron layers *(mlp_head_units) = [2048, 1024]).*

Data Augmentation

Type	*Value*
layers.Normalization()	
layers.Resizing	*(128,128)*
Layers.RandomFlip()	*(horizontal)*
layers.RandomRotation	*(factor = 0.02)*
Layers.RandomZoom()	*(*height_factor=0.2, width_factor=0.2*)*

Table of Data augmentation

Data augmentation is applied to prevent overfitting with the Caltech-256 data set. There is normalization layer that is going to normalize the pixel value in the data set images around given mean and standard deviation and then calculate this mean and standard deviation from the data set by using this *data_augmentation.layers[0].adapt(x_train)* then resize the images to 128x128 then add horizontal flipping, random rotation and random zooming.

Implementation of multilayer perceptron (MLP)

When over-writing the dense layer to add skip connection, rather than using *model.adddense,* custom *MLP* function is used to overwrite with Keras functional Api.

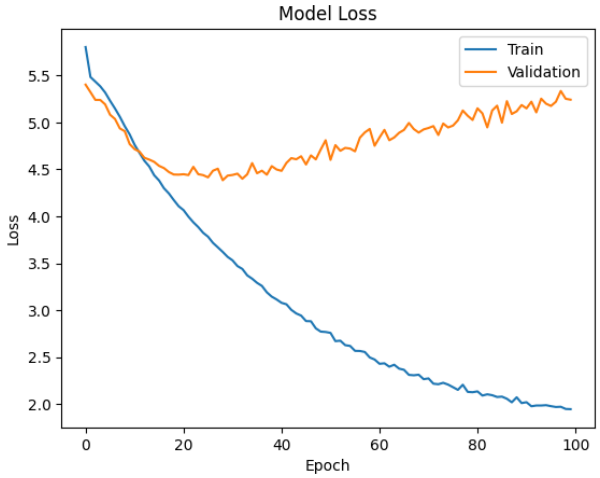
Compiling Model

Model is compiled from above hyperparameters with tensorflow-addons Adam optimizer. Loss function is *SparseCategoricalCrossentropy with* the metrics of Accuracy and top 5 accuracy. In the case of image classification of Caltech-256 dataset, it is common to have top 5-k categorical accuracy where this is similar to information retrieval. Since dataset have massive set of potential class labels (256), The top-5 accuracy metric provides an approximation of how well the model is performing, even when it makes incorrect predictions. It assesses whether the true label is among the top 5 predictions, indicating that the model has some level of understanding and is able to make reasonable guesses even if it does not predict the exact class label. For example, if there is a fighter-jet image and model is classifying as an airplane class, at least the fighter-jet is still in the top-5 prediction. This approach allows for a more comprehensive evaluation of the model's performance in scenarios where there are many possible class labels, and it is desirable to know how well the model is able to narrow down the correct class within the top 5 predictions.

Strengths	• Scalability: ViT can efficiently handle large-scale datasets, including Caltech 256, which contains a significant number of images (256 object categories with varying image sizes). • Global Context: ViT captures global image information by dividing the input image into patches and treating them as sequence data. This enables the model to learn context and relationships among different parts of the image. • Generalization: ViT has demonstrated strong generalization capabilities, achieving state-of-the-art performance on various image classification benchmarks. This indicates its ability to learn abstract representations that can be applied to different datasets, including Caltech 256.
Weaknesses	• Large Memory Requirements: ViT's reliance on self-attention mechanisms introduces scalability challenges. As the input image size increases, the memory requirements of the model also increase significantly. This can make training and inference computationally expensive, particularly for high-resolution images. • Spatial Sensitivity: ViT processes images by dividing them into fixed-size patches, disregarding the spatial information within each patch. As a result, ViT may struggle with tasks that require precise localization or fine-grained spatial understanding. • Limited Robustness to Input Variations: ViT's performance may degrade when applied to images with diverse lighting conditions, occlusions, or other types of variations that were not present in the training data. This limitation arises due to the model's reliance on large-scale datasets that may not capture all possible variations in real-world images.

Strengths of the Vision Transformer (ViT) model include its scalability for large datasets, global context understanding through attention mechanisms, and strong generalization performance. However, limitations of ViT include its lack of spatial sensitivity within patches, large memory requirements, and decreased robustness to input variations not seen in training data.

Visual Transformer Result



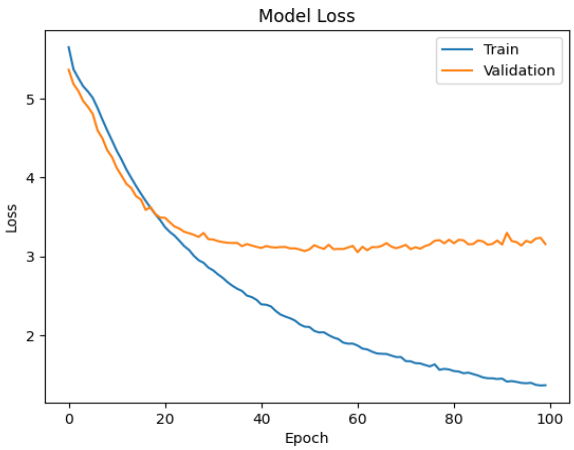
The two plots provided depict the model's accuracy and loss over epochs. Initially, the training loss steadily decreases, indicating that the model is learning and improving during training. The training accuracy and top-5 accuracy show a gradual increase as well. Similarly, the validation loss and accuracy exhibit similar trends, although the validation accuracy is slightly lower than the training accuracy.

However, after approximately 30 epochs, it becomes evident that although the training is progressing, the model's performance is not improving significantly. The accuracy and top-5 accuracy remain relatively low, and the validation accuracy does not show significant improvement. Moreover, the validation loss gradually increases while the training loss continues to decrease.

At the end of the training process, the model achieved a training accuracy of 64% and a top-5 training accuracy of 89%. However, the validation accuracy was only 18%, with a top-5 validation accuracy of 31.19%. These results strongly indicate that overfitting occurred during the training process.

The initial model performance was severely hindered by the limited dataset size of only training 7000 samples. However, upon providing a significantly larger dataset, the model's performance improved dramatically. The expanded dataset allowed the model to learn from a wider range of examples and patterns, enhancing its ability to make accurate predictions. The increased data size enabled the model

to capture more complex relationships and generalize better to unseen data, resulting in a significant boost in performance. This could not be done at initial phase due to the lack of computing power. Memory was allocated with M1 pro Device with limited 16 GB memory which was too small for 30000 sample sizes. However, with the computing power from Google Colab A100, this allowed to utilize all 30000 samples from dataset while using total 31GB of system RAM and 17.5GB of GPU RAM.



Throughout the training process, there was a gradual improvement in the model's performance. The training accuracy started at a low value of 4.09% and increased to 34.95% by the end of training. The top-5 accuracy also showed improvement, starting at 7.90% and reaching 54.68% at the end of training. Similarly, the validation accuracy started at 6.76% and increased to 35.65% by the end of training. The top-5 accuracy on the validation set started at 13.10% and reached 54.74% at the end of training. Overfitting is observed by looking at the difference train/test accuracy and loss.

This model is not a competitive result on the Caltech-256 dataset compared to other models as a Fine-tuned inception classification model where it achieved 75% accuracy. According to Keras documentation, VIT achieved better performance when it is pre-trained on jft-300 million data set, then fine-tunning it on the target dataset. This indicates that vision transformers tend to benefit more from a large-scale of data in pre-training phase. As a mitigation strategy in the absence of pre-training, one can simulate its effects through data augmentation techniques to improve performance also by adjusting other hyperparameters of the model, as mentioned in the methodology.