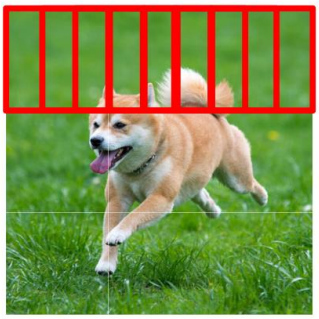
Author: Derek Walton

Vision Transformers (ViT)

Step 1 Split image into patches:

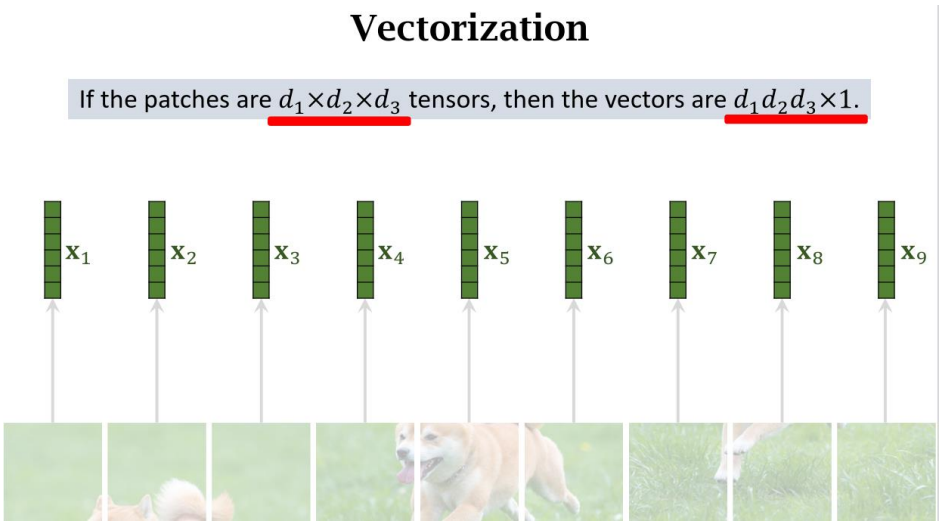
Split Image into Patches



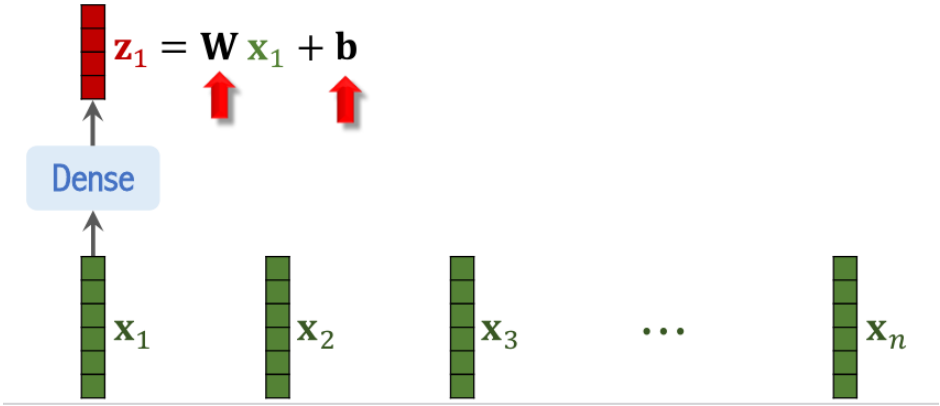
- Here, the patches do not overlap.
- The patches can overlap.
- User specifies:
  - patch size, e.g., 16×16;
  - stride, e.g., 16×16.

- The image is divided into patches of uniform size and shape, such as 14x14 or 16x16.
- To split the image into patches, two parameters are used:
  - Patch size: This determines the dimensions of each patch, such as the width and height of 14x14 or 16x16.
  - Stride size: It defines how much the "sliding window" moves each time when creating patches from the image.
    - If the stride size is smaller, it means the sliding window moves in smaller increments, resulting in a greater number of patches being generated from the image. However, this also leads to a heavier computational load when processing the image using the transformer model.

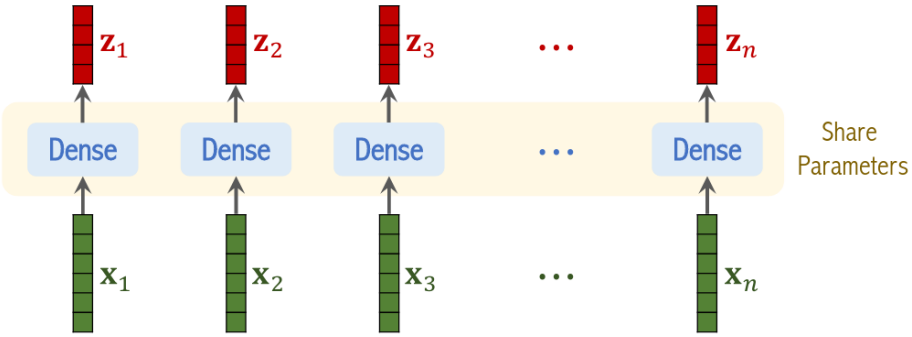
Step 2 Vectorization of Patches:



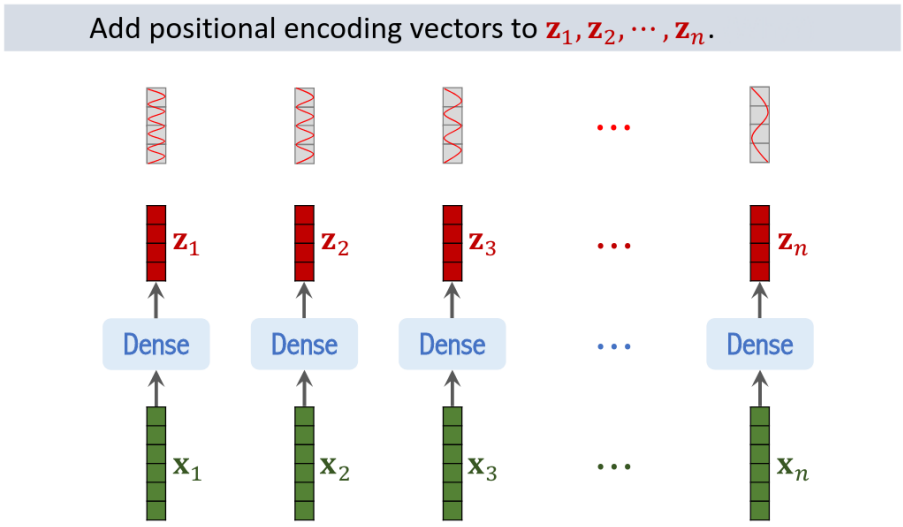
- The patches obtained from the image can be reshaped into vectors. This process involves flattening each patch, converting its 2D structure into a 1D representation.
- By reshaping the patches into vectors, we create a more convenient format for further processing. Instead of dealing with the spatial arrangement of pixels in each patch, we can treat them as sequential elements in a vector.
- This vector representation allows us to apply various operations and transformations on the patches using mathematical and computational techniques. It enables us to feed the patches into machine learning models or perform other tasks that require vectorized input.



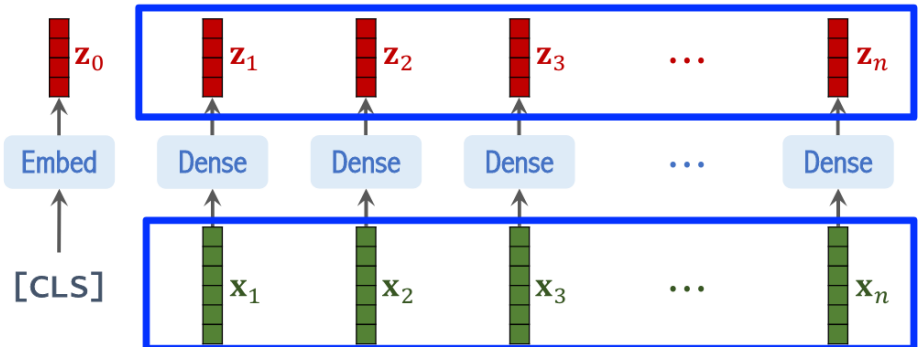
- Vectors are process through a Dense layer (Linear activation), W (matrix) and b (vector) are parameters learned from the training data



- The vectors generated from the patches do not undergo non-linear activation function; instead, a linear activation function is applied to each vector individually. The weights (W) and biases (b) are shared among all the dense layers in the transformer.

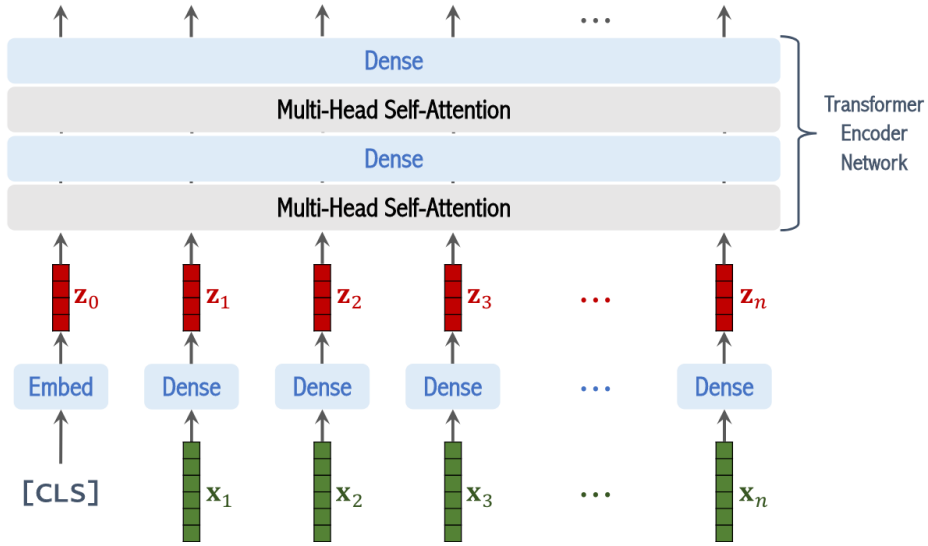


- Positional encoding is added to each Z vector
  - This allows each vector processed through the dense layer to capture both the contents and the position of the patch
  - Without positional encoding the accuracy of the transformer decreases 3% [4]
  - All the tested methods for including positional encoding produced nearly identical results, indicating that any method chosen will be sufficient for the task.[4]



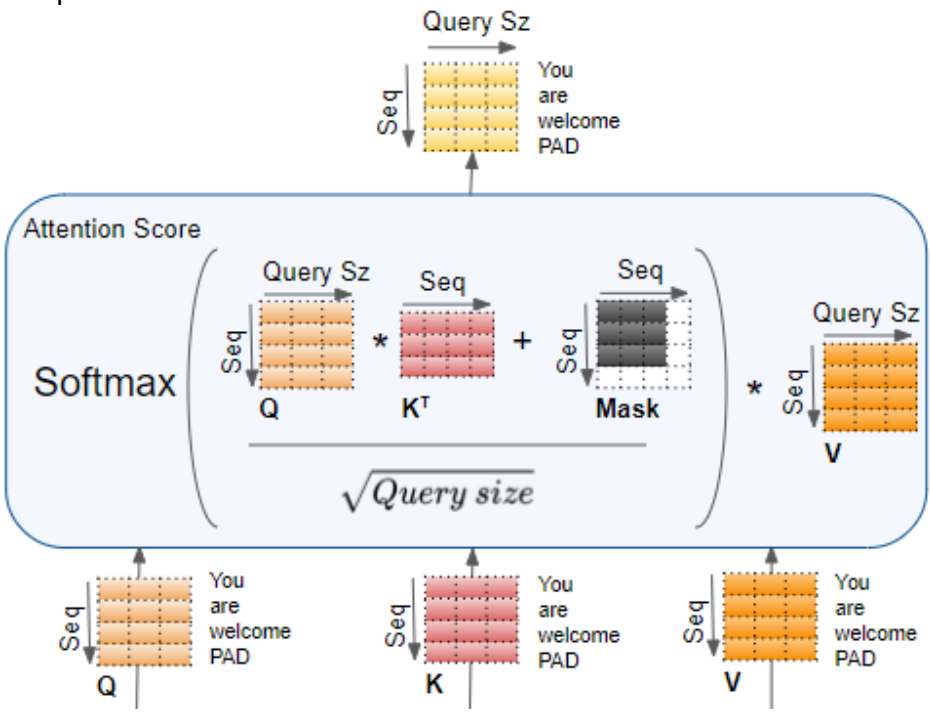
- A class token (CLS) is prepended to the beginning of the input sequence → Used for classification
    - It is randomly initialized and therefore does not contain any useful information about the image on its own
    - The token accumulates information from the other tokens in the sequence as the input sequence is processed through more layers in the transformer
- The CLS is sent to an embedded layer that yields a vector Z_0 that has the same shape as the other vectors in the input sequence

Step 3 Transformer Encoder:

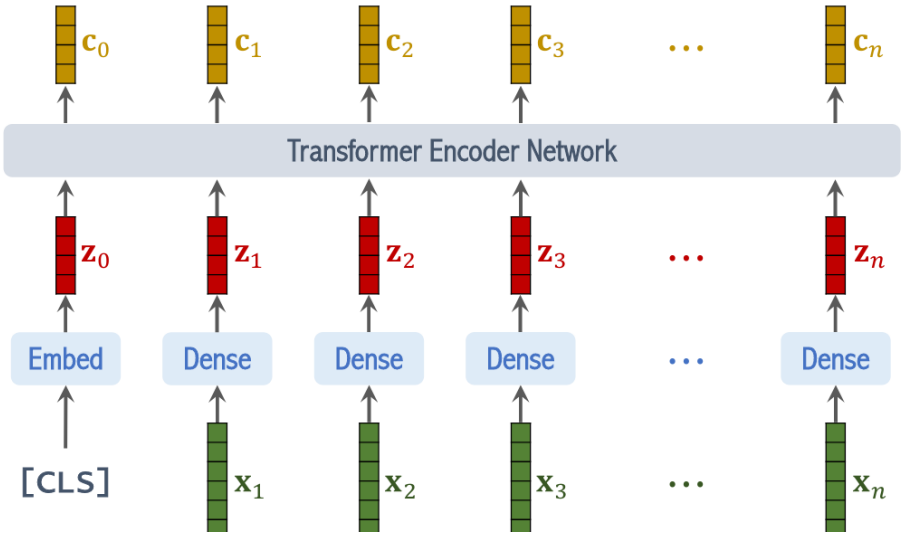


- Following linear activation all Z vectors are sent to the Transformer Encoder Network to produce n+1 context vectors
    - The network consists of an arbitrary number of Multi-Head Self-Attention and Dense Layers (i.e., when creating a visual transformer the number of layers is up to the designer of the transformer)
- Single-Head Self-Attention Layers
    - Input: Sequence of n vectors
    - Output: Sequence on n context vectors

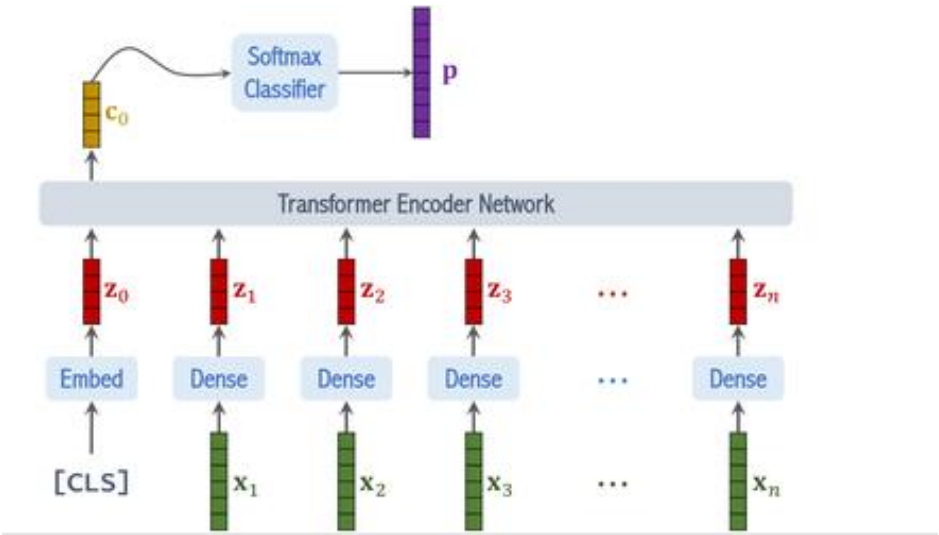
- A layer contains 3 parameter matrices (W_Q, W_k, W_V) → Elaborated on in the comparison with CNN



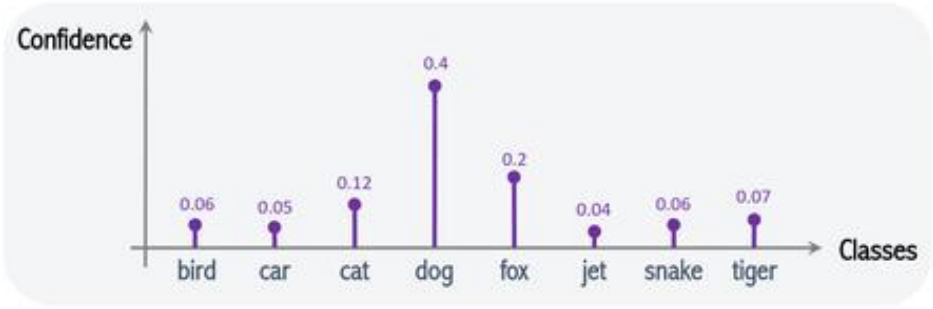
- Multi-Head Self-Attention Layers
  - Consists of L (where L is some positive integer) independently running Single-Head Self-Attention layers (i.e., don't share parameter matrices)
  - Output: Sequence of n + 1 concatenated context vectors produced from each Single-Head Self-Attention layer



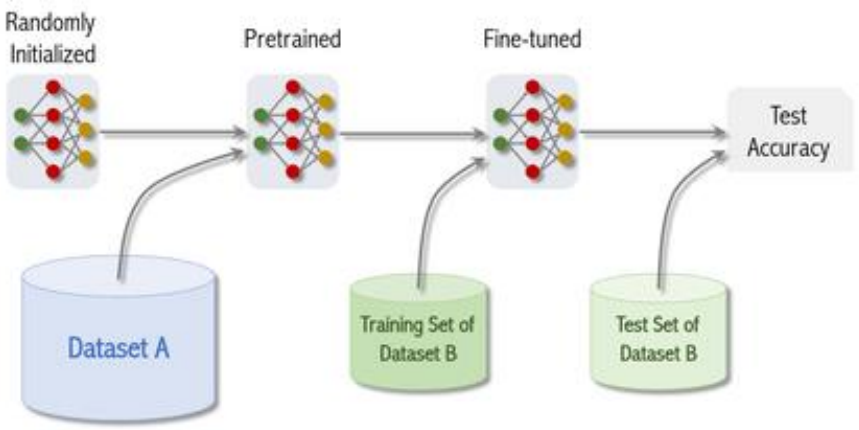
Step 4 Feed C_0 into Softmax Classifier:



- This will produce a vector p where each element in p represents a class within the dataset
    - i.e., You have a dataset containing cats and dogs p would contain 2 elements, one for cats and one for dogs



Training ViT:



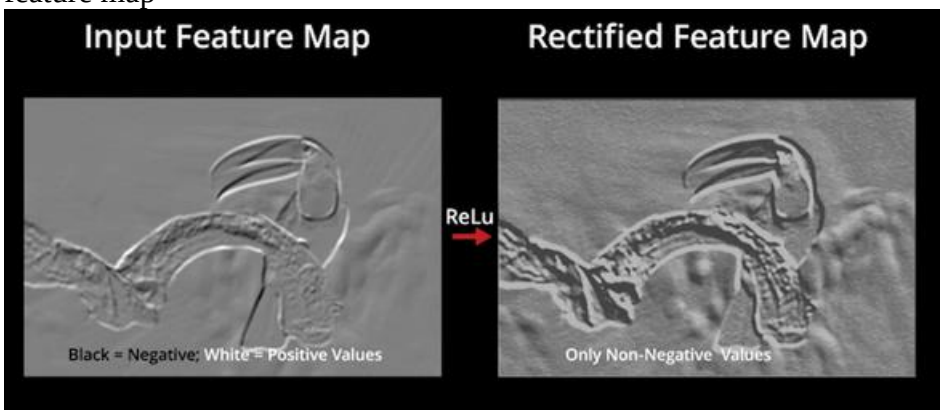
1. Randomly Initialize the Network
    1. Values to initialize the weights of the linear and self-attention layers are selected using Gaussian fill
        1. This will allow the network to explore different options when training, helping the model avoid getting stuck in symmetric or suboptimal states
        2. The biases are initialized with 0's, preventing any biases from being introduced when the training begins
    2. Other methods to randomly initialize the network have shown to produce similar results [4]
2. Train the Network on Dataset A
    1. This should be large scale dataset (30 - 100 million images minimum)
    2. Step 1 and 2 Encompass Pre-training
3. Train the Network on Dataset B
    1. Typically a smaller dataset than A, dataset B is the target dataset for the network (i.e., the dataset that you want the transformer to classify objects from)
    2. This step is referred to as fine tuning
4. Evaluate Model on Dataset B
    1. In this stage the accuracy of the network of the model is tested
        1. Dataset Split: The original dataset is divided into training, validation, and test subsets. The training subset is used for fine-tuning the model, the validation subset is used for hyperparameter tuning and model selection, and the test subset is kept separate for final evaluation.
        2. Inference: After fine-tuning, the fine-tuned transformer model is used to make predictions on the test dataset. The model takes the input data (e.g., images) and produces output predictions (e.g., class labels or bounding box coordinates) for the objects of interest.
        3. Performance Metrics: Various performance metrics are calculated to evaluate the accuracy of the model. The specific metrics depend on the

task at hand. For image classification, common metrics include accuracy, top-k accuracy (accuracy within the top k predicted classes), precision, recall, and F1 score. For object detection, metrics such as mean average precision (mAP) and intersection over union (IoU) thresholds are often used.

4. Comparison and Analysis: The performance metrics obtained from the fine-tuned transformer model are compared to those of other models or baselines to determine its effectiveness. This comparison helps understand whether the fine-tuning process has improved the model's accuracy compared to its initial state or other existing models.

CNN vs ViT

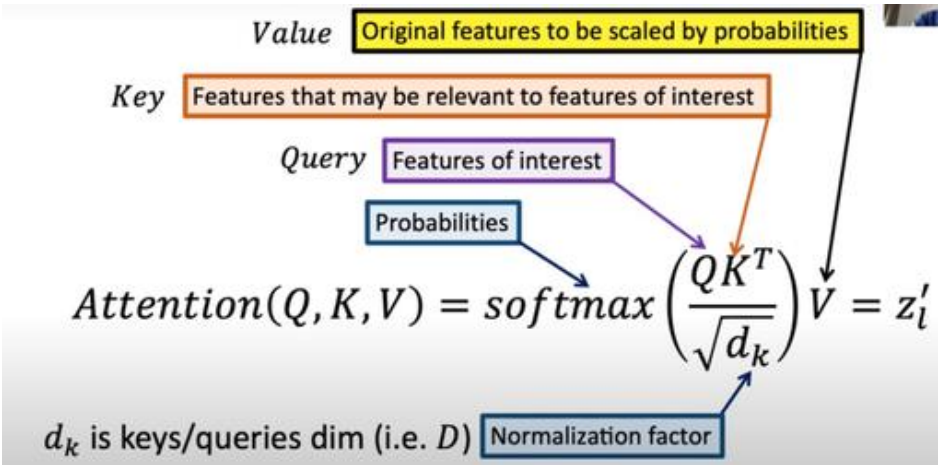
CNN

- Convolutional Layers
  - Contain filters to detect patterns in input images
    - Filters
      - Relatively small n x m matrix → values inside the matrix correspond to the pattern need for detection
      - Let's say that we have a 3x3 matrix, the filter will convolve (slide) across the image searching for the pattern in 3x3 pixel patches
    - Filters at the beginning of the convolutional layers typically detect simple geometric shapes (i.e., edges, circle, square, etc.)
    - The deeper into the layers the more sophisticated the filters become detecting things like feathers, beaks, eyes, fur, etc.
    - At the end of the convolutional layers the filter will have the ability to detect things like dogs, lizard, people, etc.
  - The output of these layers is a feature map
- ReLU Layer
  - The feature map produce in the prior layers are input into this layer
    - Non-linearity is introduced into the network here → the result is a rectified feature map
    - 

- Pooling Layer

    - This layer performs a down-sampling operation to reduce the dimensionality of the feature map
    - The pooled feature map is flattened and fed to the output layers

ViT

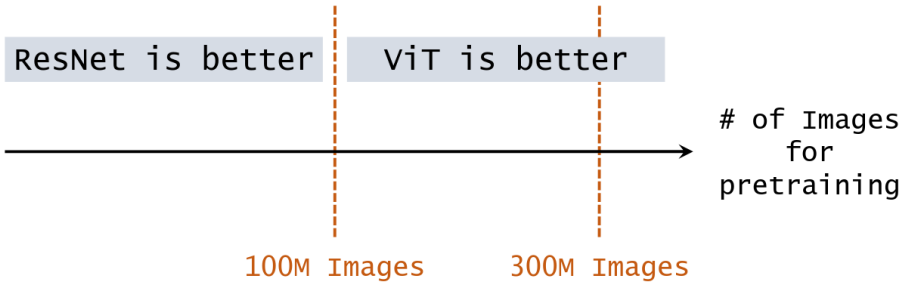
- The input image is broken down into patches
    - These patches go through a linear projection to produce Z vectors
        - Positional embedding is added to the Z vectors to indicate where each vector (patch) is located relative to the others in the sequence
    - In addition a class token (CLS) is prepended to the sequence of the Z vectors
- Encode module
    - Z vectors are fed into L (where L is a positive integer) encoder modules
    - These modules consist of Multi-Head Self Attention Layers → Used for calculating features and importance of every pixels in the image relative to the features
        - 

        - d_k is the dimension of the Z vectors
    - In addition layer normalization is performed after each Multi-Head Self Attention layer, to add normalization during the training process.

When to use ViTs

- ViTs should only be used when a very large dataset is available for training (ViTs are often referred to as data hungry)
    - Improvement over CNN models is only seen with datasets containing > 100 million images
        - However, unlike CNNs, ViTs show further improvement in prediction accuracy as the dataset increases

Image Classification Accuracies



Sources:

1).https://github.com/wangshusen/DeepLearning/blob/master/Slides/10_ViT.pdf

2). https://www.youtube.com/watch?v=HZ4j_U3FC94

3). https://www.youtube.com/watch?v=qU7wO02urYU&t=1632s

4). https://www.youtube.com/watch?v=TrdevFK_am4&t=429s

5). https://arxiv.org/pdf/2010.11929v2.pdf

6). https://huggingface.co/docs/transformers/model_doc/vit

7). https://deepganteam.medium.com/vision-transformers-for-computer-vision-9f70418fe41a

8). https://www.picsellia.com/post/are-transformers-replacing-cnns-in-object-detection#:~:text=CNNs%20are%20a%20more%20mature,information%20from%20the%20whole%20image

9). https://www.youtube.com/watch?v=hPb6A92LROc

10). https://www.youtube.com/watch?v=YRhxdVk_sIs

11). https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network