For the doing the patch embedding part which involves splitting up an image into chunks, I want to make do something very using topics learnt from the images as data and arrays lecture (1_images.jl notebook attached) where we learnt about manipulating image arrays.

Problem 3: Vision Transformers (6 pt)

So far, the best performing model we've seen for image classification has been CNNs. Recently, transformer architectures have been shown to have similar to better performance. One such model called Vision Transformer (ViT) splits up images into regularly sized patches (Dosovitskiy et al. [2020]). The patches are treated as a sequence and attention weights are learned as in a standard transformer model.

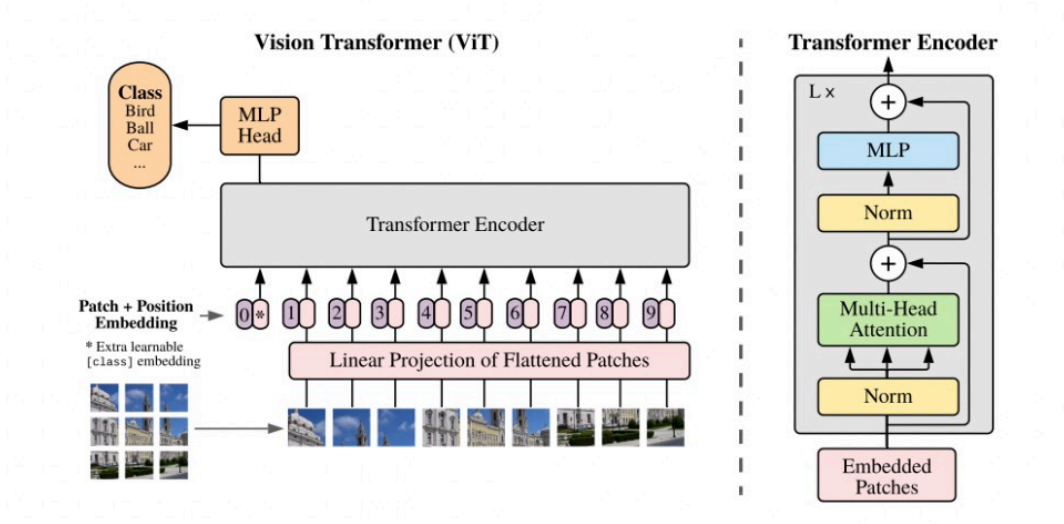

Figure 2: ViT Architecture, Figure from Dosovitskiy et al. [2020]

In detail the ViT has a few steps (see Figure 2).

- First we embed the patches using into a sequences of embeddings.
- We add a positional encoding to the embedding which captures the position of each patch in the image.
- We prepend an extra learned class embedding to our sequence and pass the entire sequence through a transformer.

- We extract the final representation of the class embedding and learn a linear layer (MLP Head) to predict the probability of each class.
- We supervise the class with cross entropy loss.

Now that we've implemented a transformer, we can use it to implement a ViT! Make sure you've already done the previous section.

(a) **(2pt)** We first implement our patch embedding. Take a look at the class `PatchEmbed`. For a given image, we want to split the image into square patches. Each patch should then be flattened and linearly projected with some weight.

For example, suppose we want embeddings of size 128. If your image is size $(3, 32, 32)$ and your patches are 4×4, you should end up with 64 patches. Flattened, each patch contains $3 * 4 * 4 = 48$ elements. We want to learn a linear projection from those 48 elements to our output dimension 128. We'll then end up with a sequence of 64 inputs of 128 elements each to pass into our transformer!

Deliverable Implement `PatchEmbed`.

Hint: Splitting up the patches manually and then using `nn.Linear` will be painful. Instead, look at `nn.Conv2d`. How can you use this to implement the patch embedding?

(b) **(1pt)** Read through the `VisionTransformer` (implemented for you). Take a look at the *positional embedding*. Positional embeddings encode the position of each element in the sequence. In this case, the positional embeddings for every position in the sequence is *learned*. However, this creates a strict limit on how many tokens can be passed to the transformer (if you only had 64 position embeddings, the positional embedding of the 65th token is undefined!)

Suppose you wanted to implement a transformer that can take arbitrarily long inputs (ignore any memory or time constraints). Describe a way to implement the positional embedding such that there is no maximum sequence size.

(c) **(1pt)** Train the Vision Transformer on CIFAR-10! We've implemented the training loop for you. Run the cells to train a model and **report your validation accuracy here** (it should be greater than 50%). This should take about 5 minutes.

(d) **(1pt)** The attention maps for transformers tell us which patch relied on which other patch. Let's take a look at the attention heatmap of the class token (averaged over all heads and layers). At a high level this can give us a sense of which parts of the image the model is relying on. We provide code to visualize this heatmap for 10 validation images.

Include the images with their heatmaps here. What do you observe about the heatmaps?

(e) **(1pt)** Previously, we have seen convolutional neural networks (CNNs) used for image classification. Answer the following questions with regards to the capabilities of CNNs and ViTs. We expect no more than a few sentences for each question.

- CNNs have inductive biases that emphasize local spatial patterns by design, while ViTs rely on global attention with limited biases. How does this difference affect their abilities, particularly on small vs. large datasets?
- CNNs capture spatial information due to their network structure, while ViTs generally rely on learned positional encodings. How do these approaches impact each model's ability to generalize to images of different sizes?