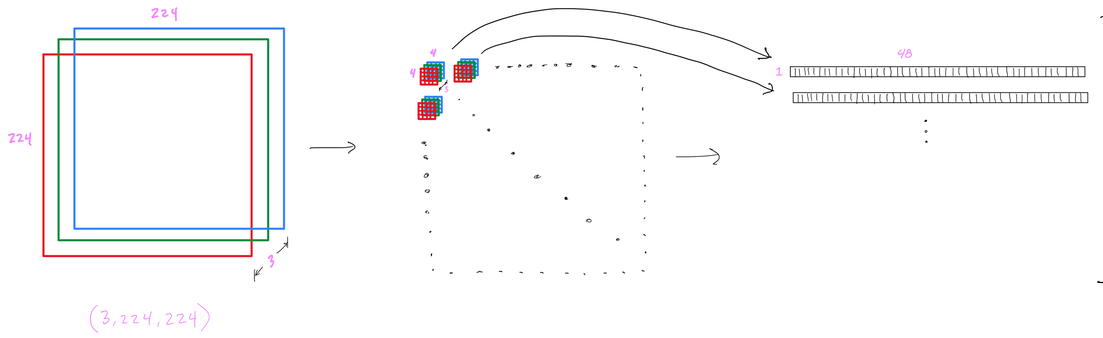
Swin Transformer AI detector

Thursday, November 16, 2023 10:15 AM



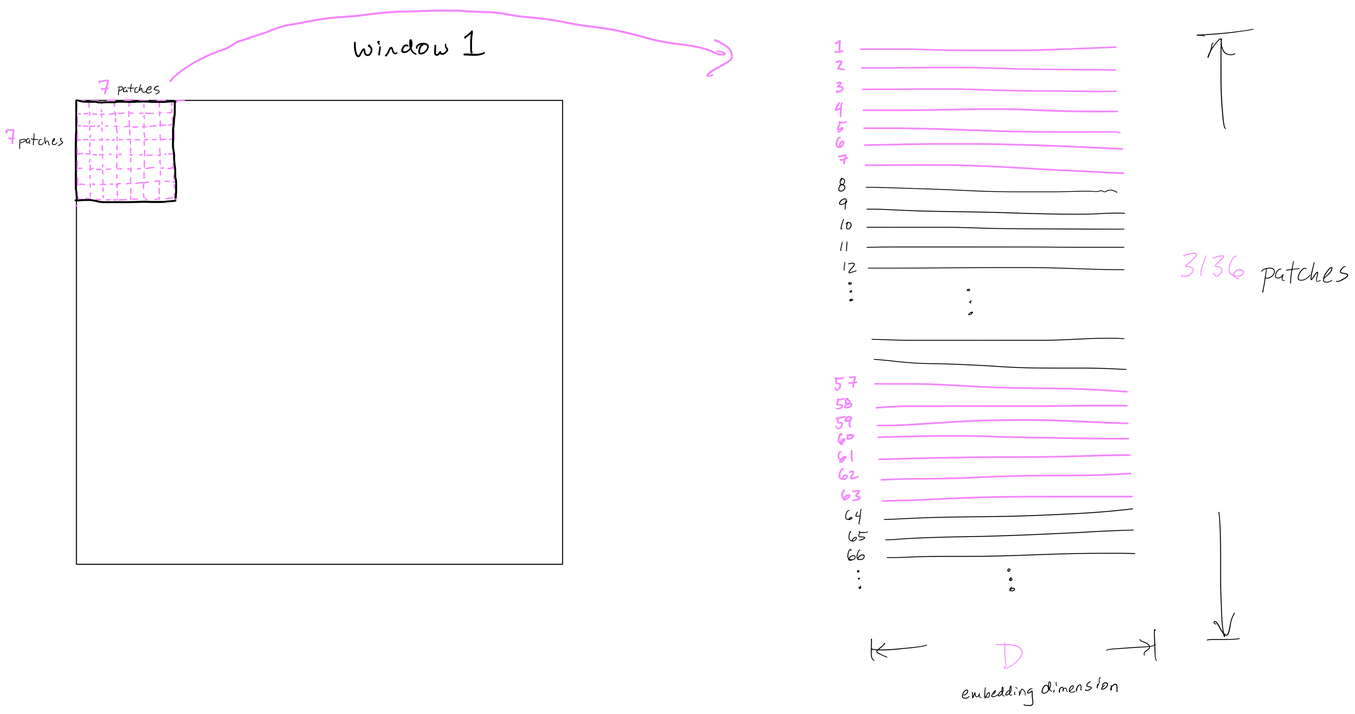
- For initial transformation stages of Swin Transformer, each image is resized to 224 width & 224 height, creating a (3,224,224) RGB tensor.
- Resized RGB tensor is divided into 4x4 patches. Each patch is still RGB at this point, so imagine 3,136 patches with shape (3,4,4)
- Each patch is then flattened to a 1 dimensional shape of (3x4x4) = [48]. This 1 dimensional tensor is what gets fed into the embedding layer

↳ This brings us to the patch embedding stage where the resulting tensor shape is [N, 3136, D], where N is the batch size, 3136 is the number of patches, and D is the embedding dimension. In other words, the linear embedding layer turns each [48] tensor into a [D] tensor. One image will be (3136, D), and a batch of images will be (N, 3136, D)

Window Partitioning & Self-Attention

Quick note about mapping: Let's consider the top left 7x7 window in the image grid. This window will contain the following patches from the flattened tensor:

- Rows 1-7 (patches from the first row of the window)
- Rows 57-63 (patches from second row of window)
- Rows 113 to 119 (patches from the third row of window)
- ... and so on until the seventh row of the window (the pattern continues skipping 49 rows each time)



- Each window is effectively a [49, D] tensor where each row corresponds to a patch
- The first step in self attention is to compute 3 matrices: Query, Key, Value. Each of which are computed by multiplying input [49, D] by learnable weight matrices. This results in 3 new matrices, each of size [49, D]
- Attention scores are then calculated by multiplying Query Matrix with transpose of Key matrix, resulting in [49x49] matrix where each element (i, j) represents attention score between patch i & patch j
- After this step, we normalize & run through softmax
- Multiply attention score matrix with Value matrix, which aggregates information from all patches in the window for each patch, weighted by their attention scores, resulting in a [49, D] tensor where each row now contains aggregated information from all other patches in window, guided by learned attention scores

Shifted Window

We will essentially take the output of self attention layer and do the same set of operations, only this time, we shift the windows right & down 3. Conceptually this will serve the purpose of accounting for the non overlapping nature of window masks, but functionally, this just looks like computing self attention again but using different indeces for each window

Final Output

After shifted window self attention, we'll typically pass tensor through some Global Average Pooling layers to bring dimensionality down from 56x56 grid of patches down to something smaller. For binary classification task, we'll eventually softmax down to 2 possible outputs: Fake, not Fake