# Convolutional Neural Networks with PEMAN

Dhushyanth S

*The use of hybrid photonic electronic neuromorphic structures to emulate the neurons of a artifical neural network, notably the multi-layered perceptron architecture has already been thoroughly explored. This report aims to expand the scope to Convolutional Neural Networks (CNNs). The use of CNNs in todays world is briefly reviewed, follwing which a method to visualize CNN as a multi-layered perceptron is introduced. The use of PEMAN to emulate the visualisation is then discussed.*

***Keywords:*** *PEMAN, CNN, Multi-layered perceptron, neuromorphic, hybrid photonic electronic neuron*

## 1 The CNN Architecture

Convolutional Neural Networks (CNNs) are a special kind of multi-layer neural network. Like almost every other neural network they are trained with a version of the back-propagation algorithm. Where they differ is in the architecture. CNNs are designed to recognize visual patterns directly from pixel images with minimal preprocessing. They can recognize patterns with extreme variability (such as handwritten characters) and with robustness to distortions and simple geometric transformations. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

## 2 Principle

CNNs for the most part, work by taking an input image and applying convolution operations over it with a kernel. The weights of the kernel are continuously updated throughout training using back-propagation.

Figure 1 shows the convolution operation. The kernel is a matrix of weights that is applied over the input image. The kernel is slid over the image and the dot product of the kernel and the image is taken. The result of the dot product is stored in the output matrix. The kernel is slid over the image by a stride. The stride is the number of pixels by which the kernel is slid over the image. The kernel is slid over the image in both the horizontal and vertical directions. The output matrix is called the feature map. The feature map is smaller in size than the input image.

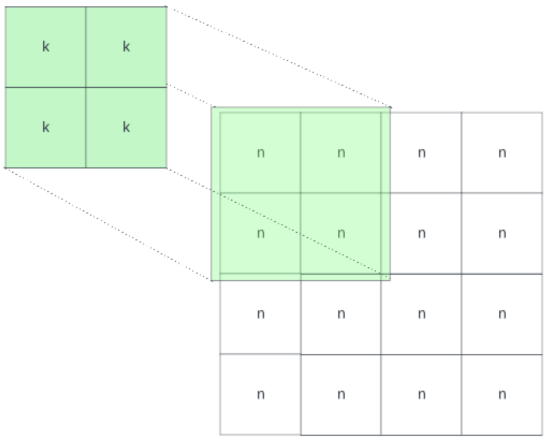

Figure 1: Convolution operation

## 3 PEMAN for CNN

The PEMAN architecture can largely be used as-is for the CNN neural networks. This can be done by visualizing the CNN network in a different manner.

From figure 1, we can see that the kernel values are being multiplied by the image pixel values and the output is being summed to give the output value for the feature map. This can be simplified as the multiplication of weights to inputs and accumulating it to get the input. Going by this analogy, we can easily see that each convolution operation applied to the image with a kernel is a neuron. The output of each neuron contributes to the feature map.

By using this train of thought, we can also safely assume that PEMAN can be used for CNN architecture. The only difference is that the PEMAN architecture will not be emulating the convolution operation directly but rather, will be executing each operation inside.

The CNN process, through the previous analogy, can be distilled down to essentially a different kind of feedforward multi-layered neural network. Thus the weight updation process itself does not change except for the fact that the weights are updated for each neuron inside the convolution operation.

A CNN model is depicted as a feedforward artificial neural network in Figure 2

The process that is being depicted in the image is as follows:

- The kernel is fit at the required position based on the stride
- Each weight in the kernel needs to be multiplied by an image pixel.

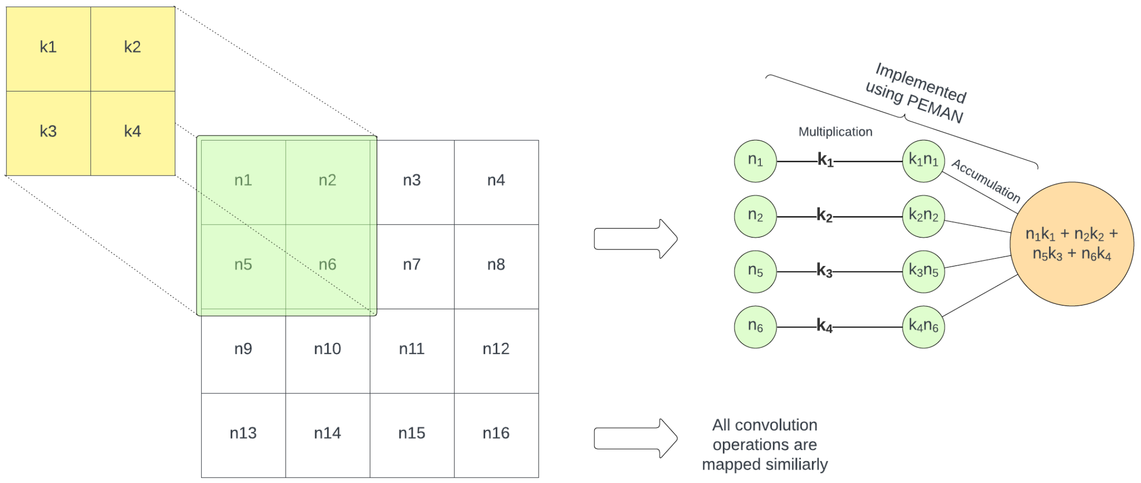

Figure 2: Realization of CNN as a more extensive ANN structure

- PEMAN structure scans the corresponding patch in the image and takes it as inputs and takes in the kernel as weights
- Each of these multiplications is done by a PEMAN neuron.
- After each multiplication, the results are accumulated together until the entire kernel is traversed.
- The output pixel value is attained, which is passed onto the next layer
- The capacitor is cleared and the kernel is moved again as per stride to repeat the process.

From the steps outlined below, we can conclude that the timing diagram for such a structure can be schematically represented as in Figure 3

Let us assume the amount of time the PEMAN structure takes to do one multiplication operation is $T_{MUL}$, and the time it takes to ensure it is ready for the next cycle is $T_S$.

Now, taking an image of $N \times N$ with a kernel size of $k \times k$, we study the total time it takes to process the entire image using this process.

The output image will be of size $(N-k+1)\times(N-k+1)$. Therefore, the total number of multiplication operations to be done will be

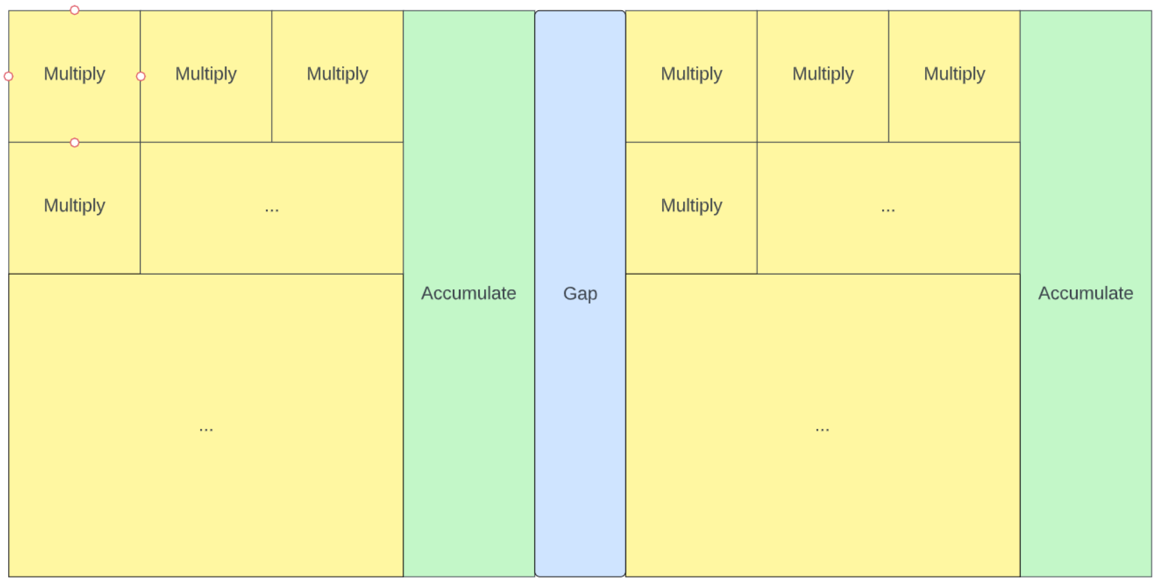

Figure 3: Timing diagram for CNN

$$\begin{aligned}&\text{Total number of multiplication operations}\\&= \text{Number of pixels in output image} \times \text{Number of pixels in kernel}\\&= (N-k+1)^2 \times k\end{aligned} \tag{1}$$

But after each time a kernel is over, the PEMAN structure will take time to reset for the next cycle which needs to be accounted for. Therefore, the total time taken to process the entire image will be

$$\begin{aligned}&T_{total}\\&= \text{Time taken for one kernel scanning} \times \text{Number of pixels in image}\\&= (T_{MUL} \times k + T_S) \times (N-k+1)^2\end{aligned} \tag{2}$$

Therefore, for a model containing multiple layers, either convolution or linear layers, the total time taken for one pass is

$$\begin{aligned} &T_{total} \\ &= \text{Time taken for one kernel scanning} \times \text{Number of pixels in image} \\ &= \sum_{n} (T_{MUL} \times k + T_S) \times (N - k + 1)^2 \end{aligned} \tag{3}$$

where n is the number of layers. For linear layers, the same formula is applicable but the kernel size of replace with 1.

This is the total time taken by one single PEMAN structure to emulate the convolution of a $k \times k$ kernel over an $N \times N$ image once. In a CNN model, there might be multiple channels per layer, multiple images per batch and multiple convolution layers too, all of which needs to be taken into consideration, as per the model requirement.

Thus we can say that CNN can be emulated by the PEMAN structure when visualized as a modified form of Multi-Layered Perceptron. This can further be studied by looking into a practical example such as the MNIST dataset. The next chapter creates a CNN-based deep learning model and implements it while simulating the PEMAN architecture to study the accuracy of the model.

## 4 Dependence of timing on kernel size

In every CNN layer, the kernel size is a very important factor. Many of the features that need to be extracted from the image are highly dependent on the kernel size. In our case, the aforementioned factors along with the speed of the model are also dependent.

We can study the dependence of the total time taken for each kernel from equation 3. The relation between kernel size and total time taken can be seen in Figure 4

As is evident from both the plot and the equation, the relation between the total time taken and the kernel size is a polynomial curve. This figure can be attributed to the fact that while kernel size increases, the number of multiplications operations increase within the kernel but the total number of times the image is sampled decreases since the kernel is large when compared to the image itself. The second inflection point is always at $N + 1$ as it depicts the maximum kernel size which we can use, that is the entire image in one pass.

Since physically, kernel size can only be positive and smaller than the image size itself, the interval $(-\infty, 0] \cup [N + 1, \infty)$ can be ignored. The remaining sample space has one inflection point, which depicts the kernel size at which the total time taken is maximum.

Upon calculating this point by differentiating and finding its roots, we get the following equation,

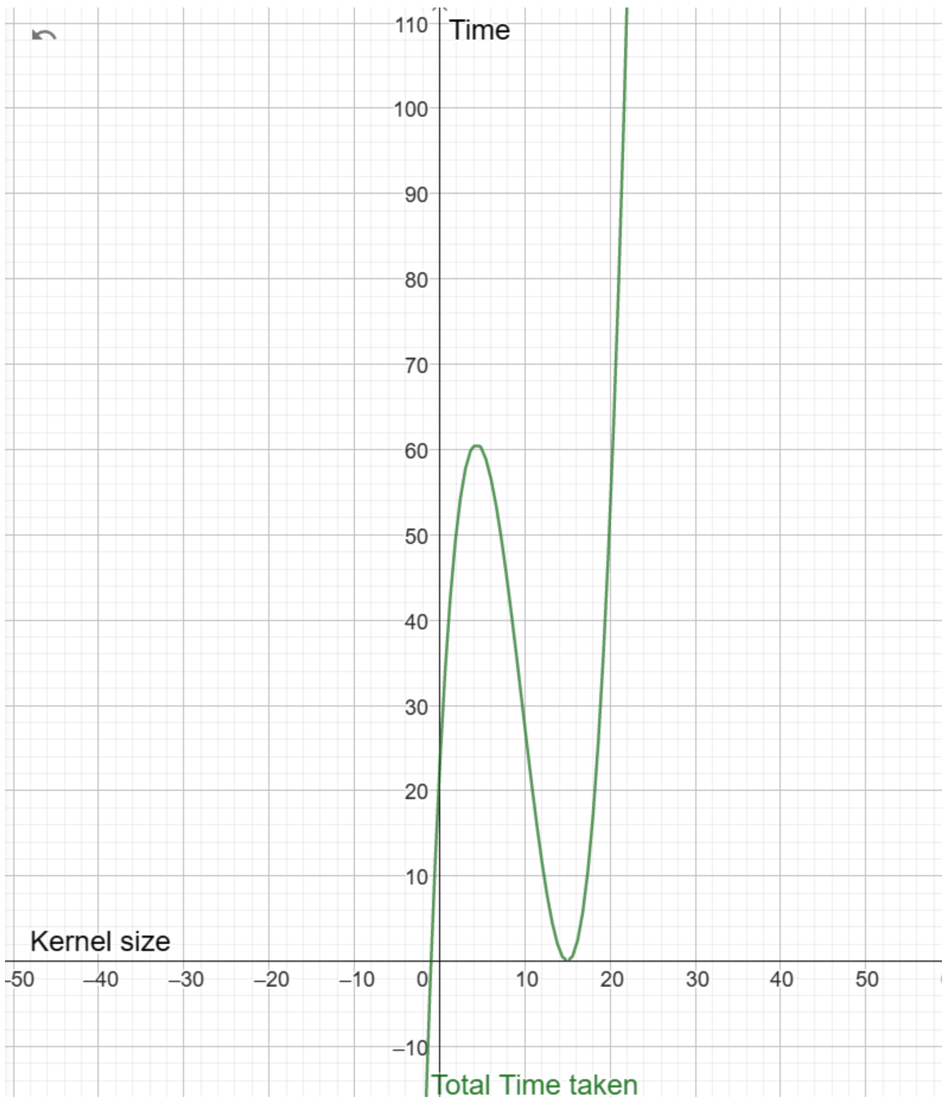

Figure 4: Relation between kernel size and total time taken

$$
\begin{aligned}
&\frac{d}{dk}\left((T_{MUL} \times k + T_S) \times (N - k + 1)^2\right) = 0 \\
&\implies (k - N - 1)\left(3T_{\mathrm{MUL}}k + 2T_{\mathrm{S}} + (-N - 1)\, T_{\mathrm{MUL}}\right) = 0 \\
&\implies k = -\frac{2T_{\mathrm{S}} + (-N - 1)\, T_{\mathrm{MUL}}}{3T_{\mathrm{MUL}}}
\end{aligned} \tag{4}
$$

At this point, the time taken for the PEMAN structure to successfully emulate the convolution operation is maximum.

## 5 Quantization

Using the architecture and emulation technique outlined in this report, all the quantization studies done for the PEMAN structure can be directly applied and utilized.

The complexity of the model is not affected by the requirement of quantization since all the multiplication and addition are handled by the PEMAN structure, which handles the quantization internally. The only change that needs to be done is the quantization of the input image, which can be done by the host processor before sending the image to the PEMAN structure.

## 6 Conclusion

In this short study, we have seen how the PEMAN structure can be used to emulate the convolution operation in a CNN. We have seen that the PEMAN structure can be efficiently used to emulate the convolution layer, which opens many possibilities for the structure. The relation between kernel size and the time taken for the convolution layer was also studied in detail. This can be used to determine the optimal kernel size for a given image and model requirements.