CS 337, Fall 2023
Convolutional Neural Networks

Scribes: Narkedamilli Harika, Nilabha Saha, Akkapally Shasmith Krishna
Rasaputra Srujana sri, Rishit Shrivastava, Harshith Sudarshanam
Edited By: Mayank Jain

September 26, 2023

Recap

Optimizers

Optimizers are used for providing adapting learning rates during training process. Different Optimizers used are:

- **SGD with momentum**: Concept of Momentum, smoothed past gradients over time
- **AdaGrad**: Adaptive learning rates per feature
- **RMSProp**: Improvement on AdaGrad
- **Adam**: Combination of SGD with Momentum and AdaGrad

1 Bias Correction in Adam

Recall that

$$s_t \leftarrow \gamma s_{t-1} + (1-\gamma) g_t \odot g_t$$
$$v_t \leftarrow \beta v_{t-1} + (1-\beta) g_t$$

Bias Correction for Adam optimizer is as follows:

$$\hat{s}_t \leftarrow \frac{s_t}{1-\gamma^t}, \quad \hat{v}_t \leftarrow \frac{v_t}{1-\beta^t}$$

With bias correction included, the weight update becomes:

$$w_t \leftarrow w_{t-1} - \frac{\eta}{\sqrt{\hat{s}_t} + \epsilon} \odot \hat{v}_t$$

Why do we need bias correction? Let us expand s_t:

$$\begin{aligned}
s_t =& (1-\gamma) g_t \odot g_t + \gamma s_{t-1} \\
=& (1-\gamma) g_t \odot g_t + \gamma(1-\gamma) g_{t-1} \odot g_{t-1} + \gamma^2 s_{t-2} \\
& \vdots \\
=& (1-\gamma) g_t \odot g_t + (1-\gamma)\gamma g_{t-1} \odot g_{t-1} + \cdots + (1-\gamma)\gamma^t s_0 \\
=& (1-\gamma) g_t \odot g_t + (1-\gamma)\gamma g_{t-1} \odot g_{t-1} + \cdots + (1-\gamma)\gamma^{t-1} s_1 \quad [\because s_0 = 0]
\end{aligned}$$

Consider the sum of the coefficients in the above equation:

$$(1-\gamma)(1+\gamma+\gamma^2+\ldots+\gamma^{t-1})$$

Firstly, note that for large values of t, we have:

$$1+\gamma+\gamma^2+\cdots+\gamma^{t-1} \approx \frac{1}{1-\gamma}$$

Thus, large values of t, the sum of the coefficients is approximately equal to 1.
However, for small values of t, we only have:

$$1+\gamma+\gamma^2+\cdots+\gamma^{t-1} = \frac{1-\gamma^t}{1-\gamma}$$

Thus, for small values of t, the sum of the coefficients only equal to $1-\gamma^t$.
To remove this bias, we use a bias correction term for s_t, which is $\hat{s}_t = s_t/(1-\gamma^t)$.

2 Convolutional Neural Networks

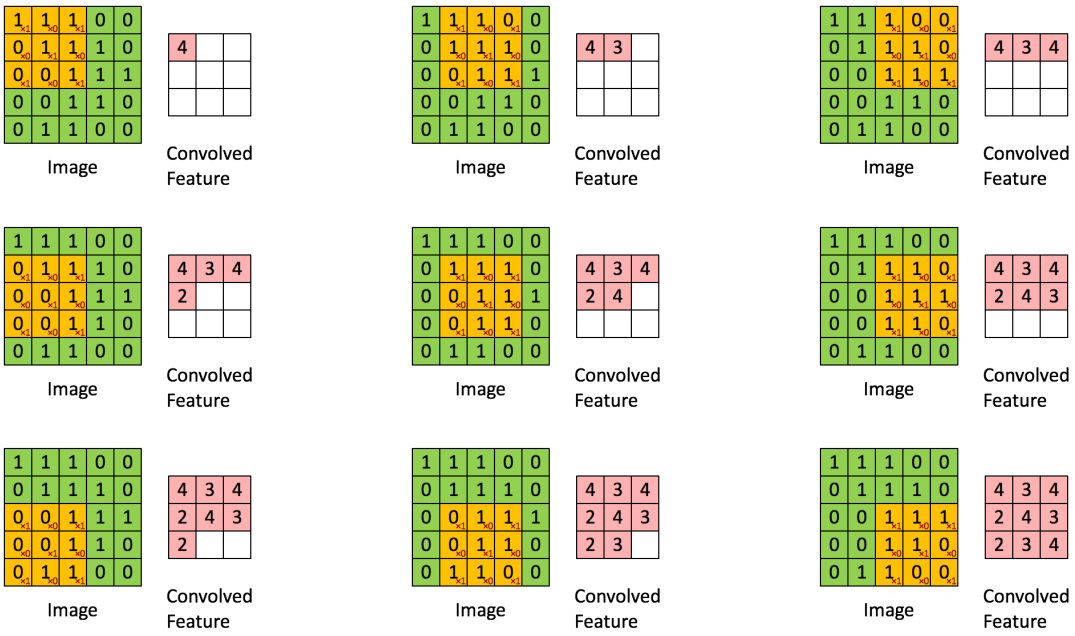
Convolutional neural networks are the neural networks which have convolution to be a basic operation on which they operate on (other than activation functions). So what difference does this make ?
Consider the problem of recognizing an object in the image, we would consider this image to be in the form of stacked pixels. In this case, we would like our model to identify or use locality that is the model should be able to learn about a pixel from its neighbouring pixels. The model should be invariant to translation, that is the position of a object in the image shouldn't matter to the model. Also, the model should be parameter efficient.
Fully connected(dense) layers have no awareness of spatial information. The key concept behind the convolutional layers is to use kernels or filters to detect spatial information. Kernels can be understood as small arrays (typically squares) of learnable weights or parameters. These filters slide across an input to detect spatial patterns in local regions. Also, the only learnable weights in this case are the weights in filter. Observing locality and translation invariance are two important features of CNNs which makes it a good choice for pattern recognition or computer vision problems.

Desired Features	Feedforward neural networks	Convolutional Neural Network
Locality	✗	✓
Translation Invariance	✗	✓
Parameter Efficient	✗	✓

Table 1: Desired Features: FFN vs CNN

3 Convolution Operation

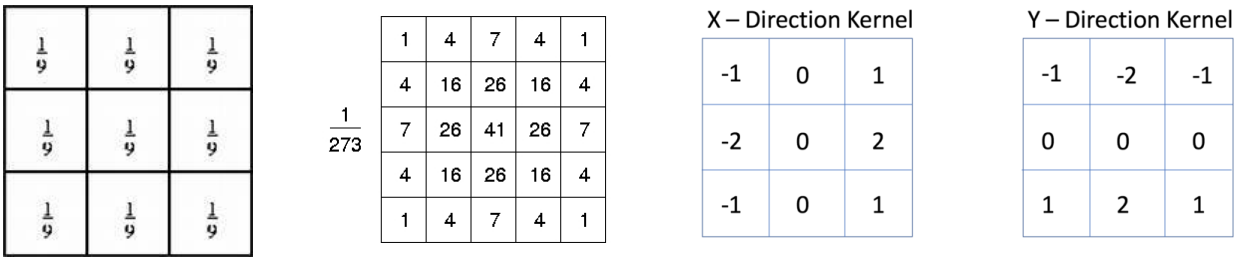
In a convolution operation, a kernel (also called a filter) is moved along the image to get convolved features. When convolving, you compute the dot product between all elements of the kernel with the value of the image underneath the kernel (in essence, taking a weighted sum over the region). This is the main operation underlying CNNs and is called convolution. (Technical Aside: This is not technically convolution, it's instead called cross-correlation.)
Here is an example below:



These filters can capture different features of the image.

- **Mean Kernels**: It averages all underlying elements of input image resulting in a blurred image. This helps retain important information and removes contrast information in image leading to a smoothed image.
- **Gaussian Kernels**: It gives more weight to pixels closer to the center of the kernel and less weight to pixels farther away. This results in a reduction of high-frequency noise and fine details in the image. It also gives a blurred image but is more effective at preserving fine details compared to mean kernel.
- **Edge Detection kernels**: It identifies abrupt changes in pixel intensities, highlighting the object boundaries and transitions in an image. Also called *Sobel filters*.

When we design filters for convolutional layers, we will have to decide filter size, stride and padding for the filter. Filter size is the size of the filter (since filters are square, the size of the filter would be the size of the square). Stride is the number of columns/rows to be left or the number of the rows/columns the filter has to be moved when the filter slides through the entire image. Stride would determine how fast the convolution is done. Padding tells how much extra pixel columns or rows (of value 0) are to be added to the image in order to change the size of the convolved image. Padding is typically used to preserve the spatial dimensions or to control the size reduction that occurs during convolution.



((a)) Mean Kernels ((b)) Gaussian Kernels ((c)) Edge Detection Kernels

Figure 2: Different Types of Filters

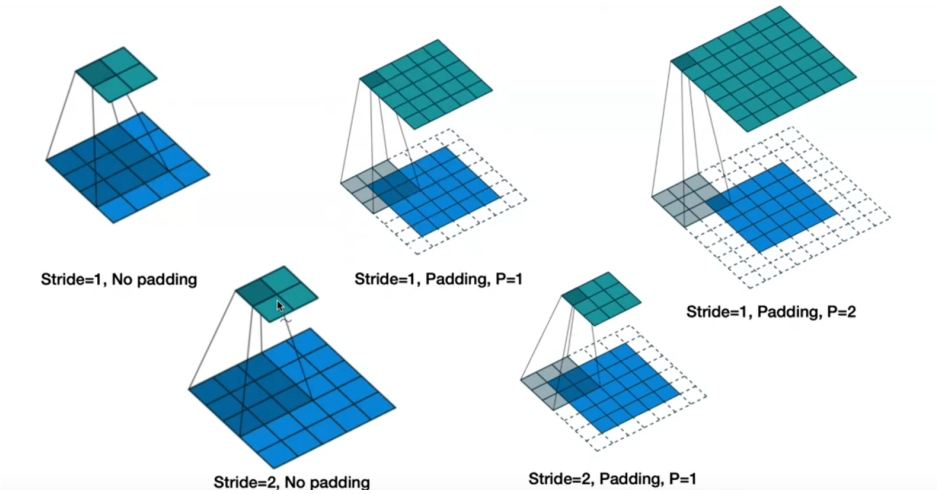


Figure 3: Stride & Padding

4 Components of CNN

4.1 Convolution Layer

In CNNs, we learn the kernel weights/matrices from data. In a convolution layer, we use multiple kernels. We fix the dimensions of each of these kernels and stack the output from these multiple kernels. Using convolution we could decrease the size of output feature map.

In the figure below, there are 3 channels/volumes in the image and we are using $5 \times 5 \times 3$ sized filter after convolving we get $28 \times 28 \times 1$ sized featured/activation map. Similarly we can use multiple filters.
In second figure we are using 6 different $5 \times 5 \times 3$ filters which results in $28 \times 28 \times 6$ sized activation map.

((a)) Single filter ((b)) Multiple filters

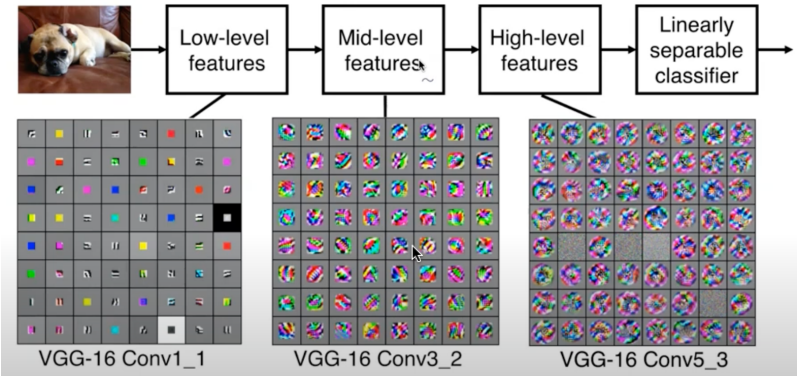
In short, a convolution layer contains multiple (say K) filters of dimension $F \times F$. It may also contain K biases. The number of parameters that have to be learned is $F \times F \times D$ parameters per filter, for K filters and K biases, thus summing up to $F \times F \times D \times K + K$ parameters.

Why do we need different layers?

- With different layers we can have different kernels focusing on different aspects of image which captures different properties that are relevant to final classification task.
- It gives flexibility to learn different properties.

4.2 What do the convolution layers learn

Each convolution layer learns some feature of the input image. The ones at the beginning tries to learn more low-level or basic features, whereas the ones at the end learn more complex or high-level features.



Then these learned high-level features from the last layer are fed into a feed forward network layer for classification.

4.3 Pooling Layer

Pooling layer is used for reducing the dimensions of feature maps. It helps in speeding up the computation and making the model robust. Max Pooling only retains feature with maximum value within the filter. It is one of the most popular pooling methods used in CNNs. Another pooling method used is Average Pooling.

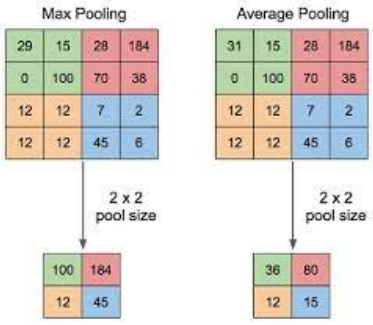


Figure 5: Pooling

NOTE: No learnable parameters are introduced by a pooling layer. Its only role is down-sampling.

5 CNN Size Arithmetic

5.1 Convolution Layers

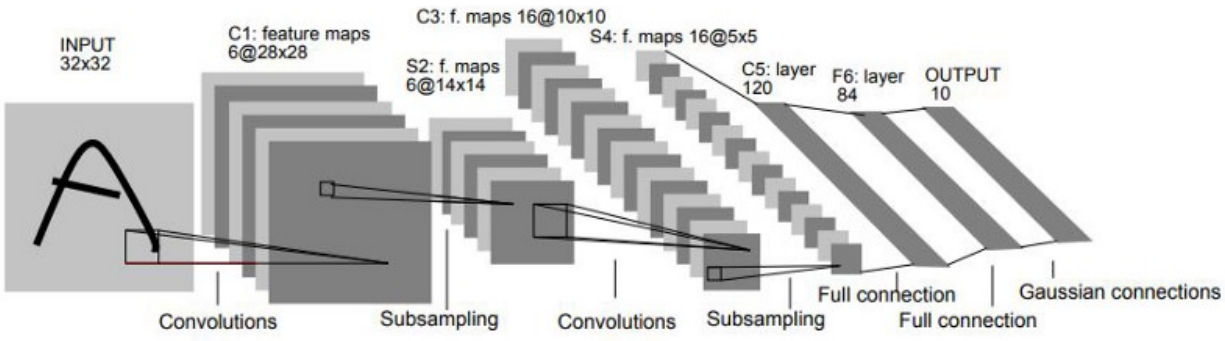
- Accepts a volume of size $W_1 \times H_1 \times D_1$.
- Requires four hyperparameters:
  - Number of filters - K
  - Their spatial extent - F
  - Stride - S
  - Amount of zero padding - P
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = \lfloor (W_1 - F + 2P)/S \rfloor + 1$
  - $H_2 = \lfloor (H_1 - F + 2P)/S \rfloor + 1$
  - $D_2 = K$

  Width and height are computed equally by symmetry.
- With parameter sharing (the same filter used for all regions in the image), we introduce $F \times F \times D_1$ weights per filter, for a total of $(F \times F \times D_1) \times K$ weights and K biases.

5.2 Pooling Layers

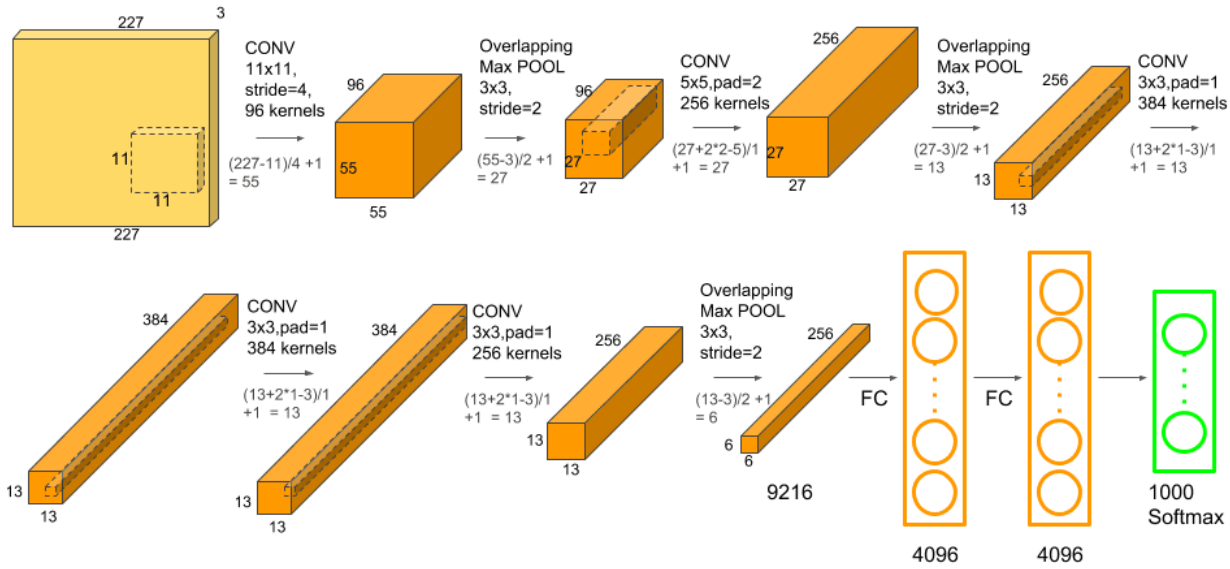
- Accepts a volume of size $W_1 \times H_1 \times D_1$.
- Requires two hyper-parameters:
  - Spatial extent - F
  - Stride - S
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = \lfloor (W_1 - F)/S \rfloor + 1$
  - $H_2 = \lfloor (H_1 - F)/S \rfloor + 1$
  - $D_2 = D_1$

6 LeNet-5 Architecture



- This was one of the first successful CNN architectures.
- It was used to classify handwritten digits.
- Led to the famous MNIST handwritten digits dataset.

7 AlexNet Architecture



- Another successful CNN architecture.
- Had better prediction accuracy than LeNet-5.