Convolutional Neural Networks, Explained

Let's build your first CNN model

Mayank Mishra
Aug 26, 2020 9 min read

Photo by Christopher Gower on Unsplash

A Convolutional Neural Network, also known as C
class of neural networks that specializes in proce

a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

_Figure 1: Representation of image as a grid of pixels (Sc

The human brain processes a huge amount of inf
second we see an image. Each neuron works in it
field and is connected to other neurons in a way t
entire visual field. Just as each neuron responds t
the restricted region of the visual field called the
the biological vision system, each neuron in a CNI
only in its receptive field as well. The layers are a
way so that they detect simpler patterns first (lin
more complex patterns (faces, objects, etc.) furth
CNN, one can enable sight to computers.

Convolutional Neural Network Archit

A CNN typically has three layers: a convolutional l
layer, and a fully connected layer.

Figure 2: Architecture of a CNN (Source)

Convolution Layer

The convolution layer is the core building block of
the main portion of the network's computational

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

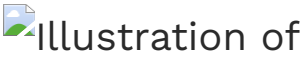


Illustration of Convolution Operation (source)

During the forward pass, the kernel slides across
width of the image-producing the image represen
receptive region. This produces a two-dimensiona
the image known as an activation map that gives
the kernel at each spatial position of the image. T
the kernel is called a stride.

If we have an input of size W x W x D and Dout nu
with a spatial size of F with stride S and amount
the size of output volume can be determined by t
formula:

$$W_{out} = \frac{W - F + 2P}{S} +$$

Formula for Convolution Layer

This will yield an output volume of size *Wout* x _Wou_t x *Dout*.

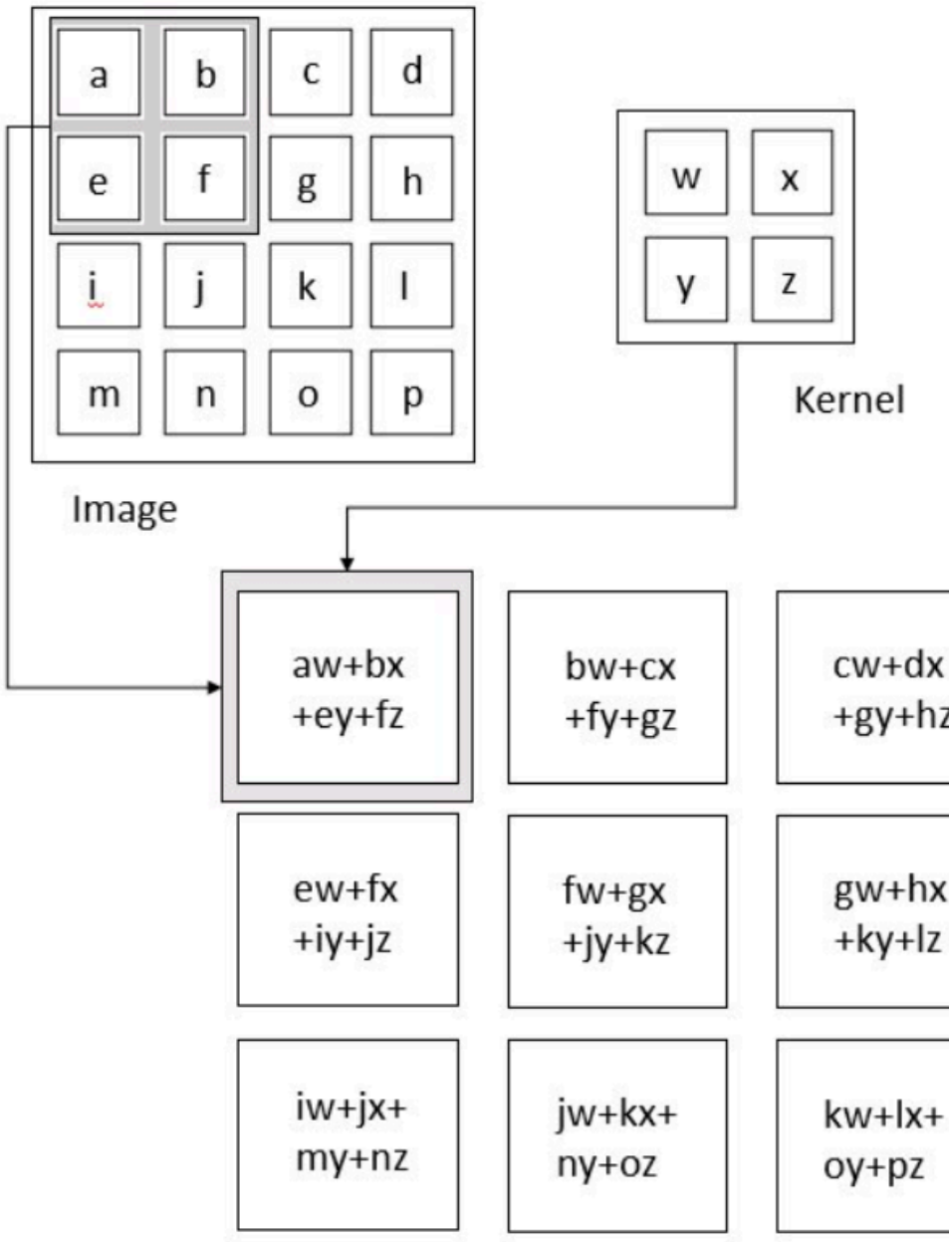


Figure 3: Convolution Operation (Source: Deep Learning by Ian Goodfellow, Yoshu

LATEST

EDITOR'S PICKS

DEEP DIVES

CONTRIBUTE

NEWSLETTER

Sign in

Contributor Portal

Motivation behind Convolution

Convolution leverages three important ideas that Computer Vision researchers: sparse interaction, and equivariant representation. Let's describe eac detail.

Trivial neural network layers use matrix multiplication by a matrix of parameters describing the interaction between the input and output unit. This means that every output unit interacts with every input unit. However, convolution neural networks have *sparse interaction*. This is achieved by making kernel smaller than the input e.g., an image can have millions or thousands of pixels, but while processing it using kernel we can detect meaningful informat
or hundreds of pixels. This means that we need t
parameters that not only reduces the memory rec
model but also improves the statistical efficiency

If computing one feature at a spatial point (x1, y1)
should also be useful at some other spatial point
means that for a single two-dimensional slice i.e.,
activation map, neurons are constrained to use th
weights. In a traditional neural network, each eler
matrix is used once and then never revisited, whil
Network has *shared parameters* i.e., for getting ou
applied to one input are the same as the weight a

Due to parameter sharing, the layers of convolutic
will have a property of *equivariance to translation*
changed the input in a way, the output will also g
same way.

Pooling Layer

The pooling layer replaces the output of the netw
locations by deriving a summary statistic of the n
helps in reducing the spatial size of the represent
decreases the required amount of computation a

pooling operation is processed on every slice of the representation individually.

There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, the most popular process is max pooling, which reports the maximum output from the neighborhood.

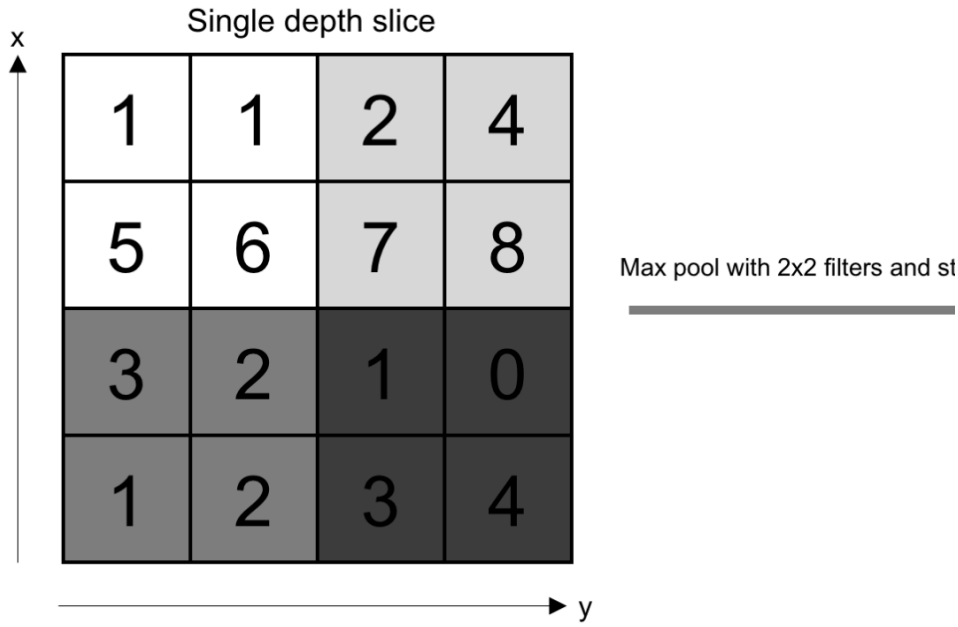


Figure 4: Pooling Operation (Source: O'Reilly Media,

If we have an activation map of size *W* x *W* x *D*, a spatial size *F*, and stride *S*, then the size of outpu determined by the following formula:

$$W_{out} = \frac{W - F}{S} + 1$$

LATEST
EDITOR'S PICKS
DEEP DIVES
CONTRIBUTE
NEWSLETTER
Sign in
Contributor Portal

Formula for Padding Layer

This will yield an output volume of size *Wout* x *Wout* x *D*.

In all cases, pooling provides some translation invariance which means that an object would be recognizable regardless of where it appears on the frame.

LATEST

EDITOR'S PICKS

DEEP DIVES

CONTRIBUTE

NEWSLETTER

Sign in

Contributor Portal

Fully Connected Layer

Neurons in this layer have full connectivity with a
preceding and succeeding layer as seen in regular
it can be computed as usual by a matrix multiplic
bias effect.

The FC layer helps to map the representation bet
the output.

Non-Linearity Layers

Since convolution is a linear operation and image
linear, non-linearity layers are often placed direct
convolutional layer to introduce non-linearity to t

There are several types of non-linear operations,
being:

1. Sigmoid

The sigmoid non-linearity has the mathematical f
$1/(1+e^{-\kappa})$. It takes a real-valued number and "squa
range between 0 and 1.

However, a very undesirable property of sigmoid is that when the activation is at either tail, the gradient becomes almost zero. If the local gradient becomes very small, then in backpropagation it will effectively "kill" the gradient. Also, if the data coming into the neuron is always positive, then the output of sigmoid will be either all positives or all negatives, resulting in a zig-zag dynamic of gradient updates for weight.

LATEST

EDITOR'S PICKS

DEEP DIVES

CONTRIBUTE

NEWSLETTER

Sign in

Contributor Portal

2. Tanh

Tanh squashes a real-valued number to the range
sigmoid, the activation saturates, but – unlike the
its output is zero centered.

3. ReLU

The Rectified Linear Unit (ReLU) has become very
few years. It computes the function $f(\kappa)=\max(0,\kappa$
the activation is simply threshold at zero.

In comparison to sigmoid and tanh, ReLU is more
accelerates the convergence by six times.

Unfortunately, a con is that ReLU can be fragile du
large gradient flowing through it can update it in s
neuron will never get further updated. However, w
this by setting a proper learning rate.

Designing a Convolutional Neural Net

Now that we understand the various components
convolutional neural network. We will be using Fa

is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes. The dataset can be downloaded here.

Our convolutional neural network has architecture as follows:

[INPUT]

→[CONV 1] → [BATCH NORM] → [ReLU] → [POOL 1]

→ [CONV 2] → [BATCH NORM] → [ReLU] → [POOL 2]

→ [FC LAYER] → [RESULT]

For both conv layers, we will use kernel of spatial stride size 1 and padding of 2. For both pooling la max pool operation with kernel size 2, stride 2, ar

LATEST

EDITOR'S PICKS

DEEP DIVES

CONTRIBUTE

NEWSLETTER

Sign in

Contributor Portal

CONV 1

Input Size (W_1 x H_1 x D_1) = 28 x 28 x 1

- **Requires four hyperparameter:**
  - **Number of kernels, k = 16**
  - **Spatial extend of each one, F = 5**
  - **Stride Size, S = 1**
  - **Amount of zero padding, P = 2**
- **Outputting volume of W_2 x H_2 x D_2**
  - **W_2 = (28 – 5 + 2 (2)) / 1 + 1 = 2**
  - **H_2 = (28 – 5 + 2 (2)) / 1 + 1 = 2**
  - **D_2 = k**

Output of Conv 1 (W_2 x H_2 x D_2) = 28 x

Calculations for Conv 1 Layer (Image by Author)

POOL 1

Input Size (W_2 x H_2 x D_2) = 28 x 28 x 16

- **Requires two hyperparameter:**
  - **Spatial extend of each one, F =**
  - **Stride Size, S = 2**
- **Outputting volume of W_3 x H_3 x D_2**
  - **W_3 = (28 – 2) / 2 + 1 = 14**
  - **H_3 = (28 – 2) / 2 + 1 = 14**

Output of Pool 1 (W_3 x H_3 x D_2) = 14 x 1

LATEST
EDITOR'S PICKS
DEEP DIVES
CONTRIBUTE
NEWSLETTER
Sign in
Contributor Portal

Calculations for Pool1 Layer (Image by Author)

CONV 2

Input Size (W_3 x H_3 x D_2) = 14 x 14 x 16

- **Requires four hyperparameter:**
  - **Number of kernels, k = 32**
  - **Spatial extend of each one, F = 5**
  - **Stride Size, S = 1**
  - **Amount of zero padding, P = 2**
- **Outputting volume of W_4 x H_4 x D_3**
  - **W_4 = (14 – 5 + 2 (2)) / 1 + 1 = 14**
  - **H_4 = (14 – 5 + 2 (2)) / 1 + 1 = 14**
  - **D_3 = k**

Output of Conv 2 (W_4 x H_4 x D_3) = 14 x 1

Calculations for Conv 2 Layer (Image by Author)

POOL 2

Input Size (W_4 x H_4 x D_3) = 14 x 14 x 32

- **Requires two hyperparameter:**
  - **Spatial extend of each one, F = 2**
  - **Stride Size, S = 2**
- **Outputting volume of W_5 x H_5 x D_3**
  - **W_5 = (14 – 2) / 2 + 1 = 7**
  - **H_5 = (14 – 2) / 2 + 1 = 7**

Output of Pool 2 (W_5 x H_5 x D_3) = 7 x 7 x

Calculations for Pool2 Layer (Image by Author)

FC Layer

Input Size (W_5 x H_5 x D_3) = 7 x 7 x 32

Output Size (Number of Classes) = 10

Size of Fully Connected Layer (Image by Author)

Code snipped for defining the convnet

```
class convnet1(nn.Module):
    def __init__(self):
        super(convnet1, self).__init__()

        # Constraints for layer 1
        self.conv1 = nn.Conv2d(in_channels=1, out_c
        self.batch1 = nn.BatchNorm2d(16)
```

LATEST
EDITOR'S PICKS
DEEP DIVES
CONTRIBUTE
NEWSLETTER
Sign in
Contributor Portal

```
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2) #default stride is equiva

        # Constraints for layer 2
        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_siz
        self.batch2 = nn.BatchNorm2d(32)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2)

        # Defining the Linear layer
        self.fc = nn.Linear(32*7*7, 10)

    # defining the network flow
    def forward(self, x):
        # Conv 1
        out = self.conv1(x)
        out = self.batch1(out)
        out = self.relu1(out)

        # Max Pool 1
        out = self.pool1(out)

        # Conv 2
        out = self.conv2(out)
        out = self.batch2(out)
        out = self.relu2(out)

        # Max Pool 2
        out = self.pool2(out)
```

LATEST

EDITOR'S PICKS

DEEP DIVES

CONTRIBUTE

NEWSLETTER

Sign in

Contributor Portal

```
        out = out.view(out.size(0), -1)
        # Linear Layer
        out = self.fc(out)

        return out
```

We have also used batch normalization in our network, which saves us from improper initialization of weight matrices forcing the network to take on unit Gaussian distr for the above-defined network is available here. W using cross-entropy as our loss function and the A with a learning rate of 0.001. After training the mo 90% accuracy on the test dataset.

Applications

Below are some applications of Convolutional Neu today:

1. Object detection: With CNN, we now have sop like R-CNN, Fast R-CNN, and Faster R-CNN th predominant pipeline for many object detecti deployed in autonomous vehicles, facial detec
2. Semantic segmentation: In 2015, a group of re Hong Kong developed a CNN-based Deep Pars incorporate rich information into an image se Researchers from UC Berkeley also built fully networks that improved upon state-of-the-ar segmentation.

3. Image captioning: CNNs are used with recurrent neural networks to write captions for images and videos. This can be used for many applications such as activity recognition or describing videos and images for the visually impaired. It has been heavily deployed by YouTube to make sense to the huge number of videos uploaded to the platform on a regular basis.

References

1. Deep Learning by Ian Goodfellow, Yoshua Ben; Courville published by MIT Press, 2016
2. Stanford University's Course – CS231n: Convo Network for Visual Recognition by Prof. Fei-Fe Johnson, Serena Yeung
3. https://datascience.stackexchange.com/quest ce-of-activation-functions-in-neural-network
4. https://www.codementor.io/james_aka_yale/c neural-networks-the-biologically-inspired-mc
5. https://searchenterpriseai.techtarget.com/defi l-neural-network

WRITTEN BY

Mayank Mishra

See all from Mayank Mishra

Topics:

Cnn | Computer Vision | Convolution Network | Fashionmnist | Writing Nn

Share this article:

Related Articles

DEEP LEARNING

Speeding Up the Vision Transformer with BatchNorm

How integrating Batch Normalization in an encoder-only Transformer architecture can lead to reduced training time...

Anindya Dey, PhD
August 6, 2024 28 min read

DATA SCIENCE

How to read a bottle using c (Part 3)

Welcome back tc As a reminder, in

Antonin Leroy
March 28, 2022 6

MACHINE LEARNING

AI Got Your Back Segmented (PyTorch)

UNet segmentation for spinal columns

Mazi Boustani
January 6, 2022 8 min read

DEEP LEARNING

Semantic Seg Imagery captu using Differer Approaches

Implementing co architecture from semantic segmer imagery...

Ibrahim Kovan

January 5, 2022 9 min read

DEEP LEARNING

GLIP: Introducing Language-Image Pre-Training to Object Detection

Grounded Language-Image Pre-training by L. H. Li et. al.

Sascha Kirch

September 1, 2023 1 min read

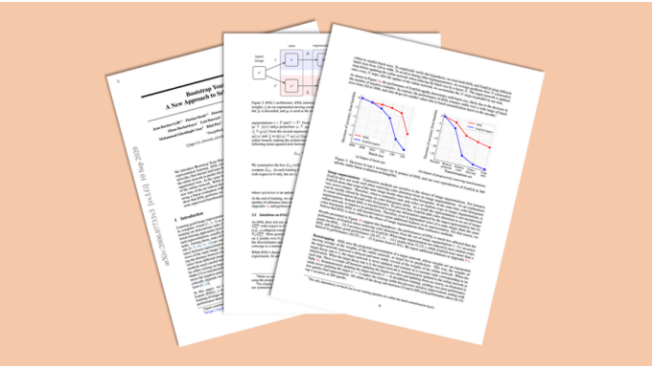
DEEP LEARNING

Segment Anyt Segmentation Objects

Segment Anythin

Sascha Kirch

September 14, 2023



DEEP LEARNING

BYOL -The Alternative to Contrastive Self-Supervised Learning

Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning by J. Grill et. al.

Sascha Kirch

September 7, 2023 1 min read

Your home for data science and Al. The world's leading publicati
analytics, data engineering, machine learning, and artificial intel

© Insight Media Group, LLC 2025

Subscribe to Our Newsletter