**Convolution:**

2D Convolution:

Zero Padding:

テキスト

自動的に生成された説明

Step 1: Matrix inversion

Step 2: Slide the kernel over the image and perform MAC operation at each instant

• 2-D discrete convolutions (N = 2),

• square inputs (i1 = i2 = i),

• square kernel size (k1 = k2 = k),

• same strides along both axes (s1 = s2 = s),

• same zero padding along both axes (p1 = p2 = p)

時計と文字の加工写真

低い精度で自動的に生成された説明

2D convolution using a kernel size of 3, stride of 1 and padding

図形

自動的に生成された説明

2D convolution with no padding, stride of 2 and kernel of 3

黒い背景と白い文字のロゴ

低い精度で自動的に生成された説明

A (half) padded convolution will keep the spatial output dimensions equal to the input.

The needed parameters for such a layer can be calculated by I\*O\*K

**Transposed convolution:**

Transposed 2D convolution with no padding, stride of 2 and kernel of 3

図形 が含まれている画像

自動的に生成された説明

The need for transposed convolutions generally arises from the desire to use a transformation going in the opposite direction of a normal convolution, i.e., from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution.

テキスト, 手紙

自動的に生成された説明

**Up Sampling:**

The Upsampling layer is a simple layer with no weights that will double the dimensions of input and can be used in a generative model when followed by a traditional convolutional layer.

テーブル が含まれている画像

自動的に生成された説明

**Dilated Convolution (a.k.a. atrous convolutions):**

2D convolution using a 3 kernel with a dilation rate of 2 and no padding

黒い背景と白い文字のロゴ

低い精度で自動的に生成された説明

Dilated convolutions “inflate” the kernel by inserting spaces between the kernel elements. The dilation “rate” is controlled by an additional hyperparameter

d. Implementations may vary, but there are usually d−1 spaces inserted between

kernel elements such that d = 1 corresponds to a regular convolution.

Dilated convolutions are used to cheaply increase the receptive field of output units without increasing the kernel size, which is especially effective when multiple dilated convolutions are stacked one after another.

A kernel of size k dilated by a factor d has an effective size k’ = k + (k − 1)(d − 1).

テーブル

中程度の精度で自動的に生成された説明

**1D Convolution:**

Conv1D is widely applied on sensory data, and accelerometer data is one of it.

テーブル

自動的に生成された説明

**3D Convolution:**

In Conv3D, the kernel slides in 3 dimensions,

ダイアグラム

自動的に生成された説明

Conv3D is mostly used with 3D image data. Such as Magnetic Resonance Imaging (MRI) data. MRI data is widely used for examining the brain, spinal cords, internal organs and many more. A Computerized Tomography (CT) Scan is also an example of 3D data, which is created by combining a series of X-rays image taken from different angles around the body. We can use Conv3D to classify this medical data or extract features from it.

**Resnet Architecture:**

Deep networks are hard to train because of the vanishing gradient problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

ダイアグラム

自動的に生成された説明

ダイアグラム, 箱ひげ図

自動的に生成された説明

ダイアグラム

自動的に生成された説明

**ResNeXt Architecture:**

Xie et al. proposed a variant of ResNet that is codenamed ResNeXt

ダイアグラム

自動的に生成された説明

it is very similar to the Inception module, they both follow the split-transform-merge paradigm, except in this variant, the outputs of different paths are merged by adding them together, while in Inception-Net they are depth-concatenated.

**DenseNet Architecture:**

Huang et al. [9] proposed a novel architecture called DenseNet that further exploits the effects of shortcut connections — it connects all layers directly with each other. In this novel architecture, the input of each layer consists of the feature maps of all earlier layer, and its output is passed to each subsequent layer. The feature maps are aggregated with depth-concatenation.

ダイアグラム, 設計図

自動的に生成された説明

Other than tackling the vanishing gradients problem, the authors argue that this architecture also encourages feature reuse, making the network highly parameter-efficient.

**Inception Net Architecture:**

**Inceptionv1:**

It performs convolution on an input, with 3 different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module.

ダイアグラム

自動的に生成された説明

Deep neural networks are computationally expensive. To make it cheaper, the authors limit the number of input channels by adding an extra 1x1 convolution before the 3x3 and 5x5 convolutions. Though adding an extra operation may seem counterintuitive, 1x1 convolutions are far cheaper than 5x5 convolutions, and the reduced number of input channels also help. Do note that however, the 1x1 convolution is introduced after the max pooling layer, rather than before.

ダイアグラム

自動的に生成された説明

Further, 5x5 convolutions can be reduced to two 3x3 convolutions thus reducing the number of parameters.

**Graph Convolution Network:**

The major difference between CNNs and GNNs is that CNNs are specially built to operate on regular (Euclidean) structured data, while GNNs are the generalized version of CNNs where the numbers of nodes connections vary and the nodes are unordered (irregular on non-Euclidean structured data).

GCNs themselves can be categorized into 2 major algorithms, **Spatial Graph Convolutional Networks** and **Spectral Graph Convolutional Network**s.

Review: <https://arxiv.org/pdf/1901.00596.pdf>

Object Detection:

Yolo:

Yolov3: <https://arxiv.org/pdf/1804.02767.pdf>

Yolov2: <https://arxiv.org/pdf/1612.08242.pdf>

Yolo: <https://arxiv.org/pdf/1506.02640.pdf>

Faster RCNN:

Paper: <https://arxiv.org/pdf/1506.01497.pdf>

PAFNet:

[PAFNet: An Efficient Anchor-Free Object Detector Guidance](https://paperswithcode.com/paper/pafnet-an-efficient-anchor-free-object)

Semantic Segmentation:

Mask RCNN:

Paper: <https://arxiv.org/pdf/1703.06870.pdf>

DeepLabv3

DeepLab Paper: <https://arxiv.org/pdf/1606.00915.pdf>

DeepLabv3: <https://arxiv.org/pdf/1706.05587.pdf>

Generative Adversarial Network (GAN):

Anime GAN: <https://arxiv.org/pdf/1708.05509.pdf>

Self-Attention GAN: <https://arxiv.org/pdf/1805.08318.pdf>

Attention is all you need: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>

Auto-encoder:

Paper: <https://arxiv.org/pdf/2003.05991.pdf>

RNN, LSTM, GRU

Pytorch Code: <https://www.kaggle.com/andradaolteanu/pytorch-rnns-and-lstms-explained-acc-0-99>

<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

Few Shot Learning (FSL)

Triplet Loss

Multi Instance Learning (MIL):

論文：<https://arxiv.org/pdf/1801.04264v3.pdf>

Code: <https://github.com/WaqasSultani/AnomalyDetectionCVPR2018>

Deep Affinity Network (DAN):

Deep Belief Network (DBN):

FlowNet2.0:

論文: <https://arxiv.org/pdf/1612.01925.pdf>

動画: <https://www.youtube.com/watch?v=JSzUdVBmQP4>

Code: <https://github.com/NVIDIA/flownet2-pytorch>

OpenPose:

論文: <https://arxiv.org/pdf/1812.08008.pdf>

Code: <https://github.com/CMU-Perceptual-Computing-Lab/openpose>

Action Recognition:

**Temporal 3D Convnets:**

Review: <https://arxiv.org/pdf/1711.08200.pdf>

I3D CNN:

ダイアグラム

自動的に生成された説明

SlowFast:

論文：<https://arxiv.org/pdf/1812.03982.pdf>

動画：<https://www.youtube.com/watch?v=jt3axjinqIM>

<https://www.youtube.com/watch?v=Flm-kkCqACM>

Code: <https://github.com/facebookresearch/SlowFast>

Dataset: Kinetics, Charades and AVA

Kinetics: <https://deepmind.com/research/open-source/kinetics>

Charades: <http://vuchallenge.org/charades.html>

Keras implementation of Action Recognition

<https://www.youtube.com/watch?v=EliLhaAf-So>