A General Overview of 3D Convolution

With analysis of 5th and 6th papers.

Very brief overview of Convolution

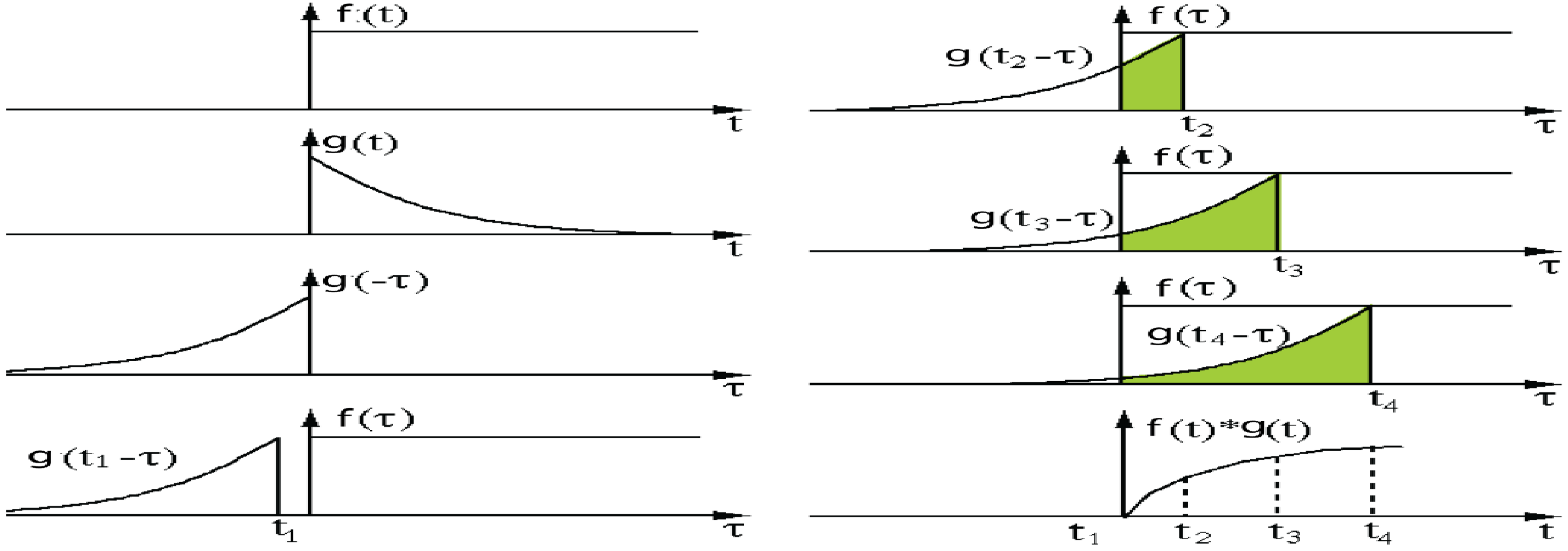
Before getting into 3D Convolution

What is Convolution?

The convolution operation or the faulting operation of two functions F and G is defined as:-

(F*G)(t)=∫F(t-x)G(x)dx(from 0 to t)

Which if pictorized is the simple integration(addition in case of discrete pixels) of the product of two functions of which one is just reversed and shifted wrt to the horizontal axis.A pictorial representation is provided below:- (Here x is replaced with tau and G is shifted instead of F).Normally,G stands for the Image and F stands for the Kernel(which moves over the Image) and convolutes over it.



The concept of convolution in Deep Learning

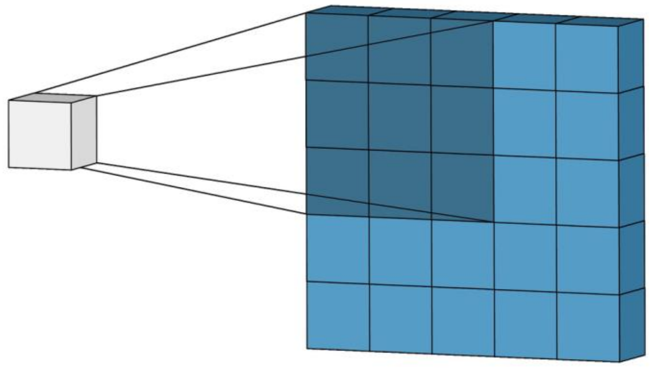
When the question comes about how the computer sees,to elaborate the process of computer vision ,the convolution concept comes along with it. .When a computer sees an image that image is become numbers of pixels for the computer. For image classification,extracting features and learning them is the most essential steps to perform.Neural Networks allow to learn those features from visual data if the data is cleverly constructed.

The basic idea of convolution is to connect the input patches to a single neuron in hidden layer.

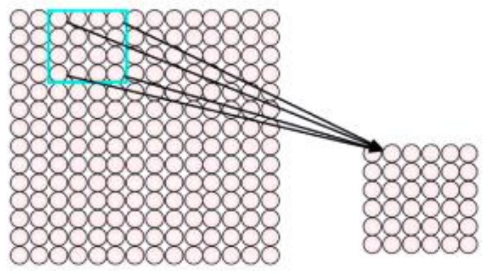
We use small patch to define connections across the entire input along with the principle of connecting patches in the input layer in single neurons and in the subsequent layer by simply using the sliding patch window across the input image

Deep learning models are a class of machines that can learn a hierarchy of features by building high level features from low-level ones.

The CNNs are a type of deep models in which trainable filters and local neighbor--hood pooling operations are applied alternatively on the raw input image and resulting in a hierarchy of increasingly complex features.

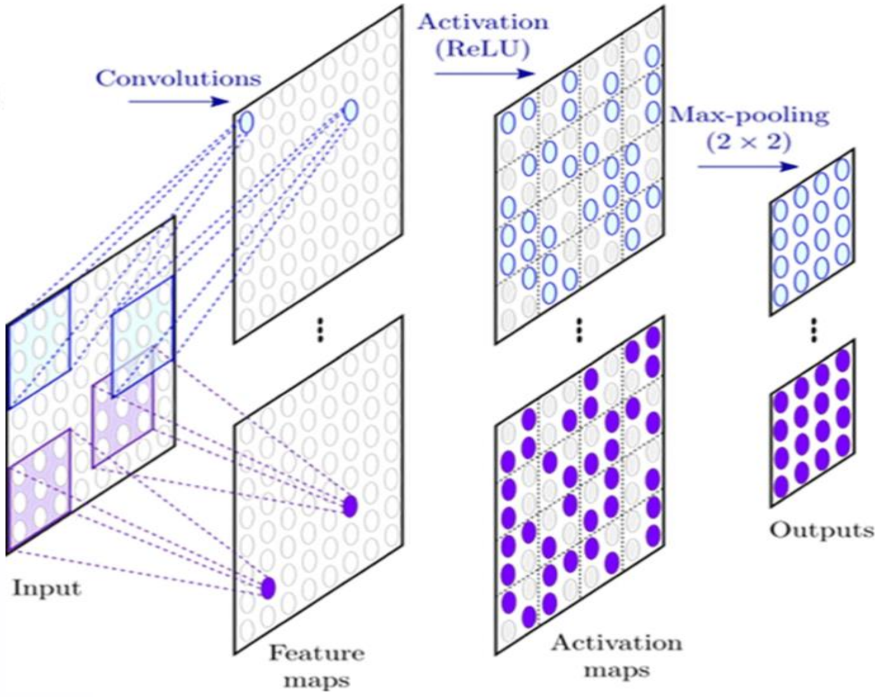


Patchy operation



- Filter of size 4x4 : 16 different weights
- Apply this same filter to 4x4 patches in input
- Shift by 2 pixels for next patch

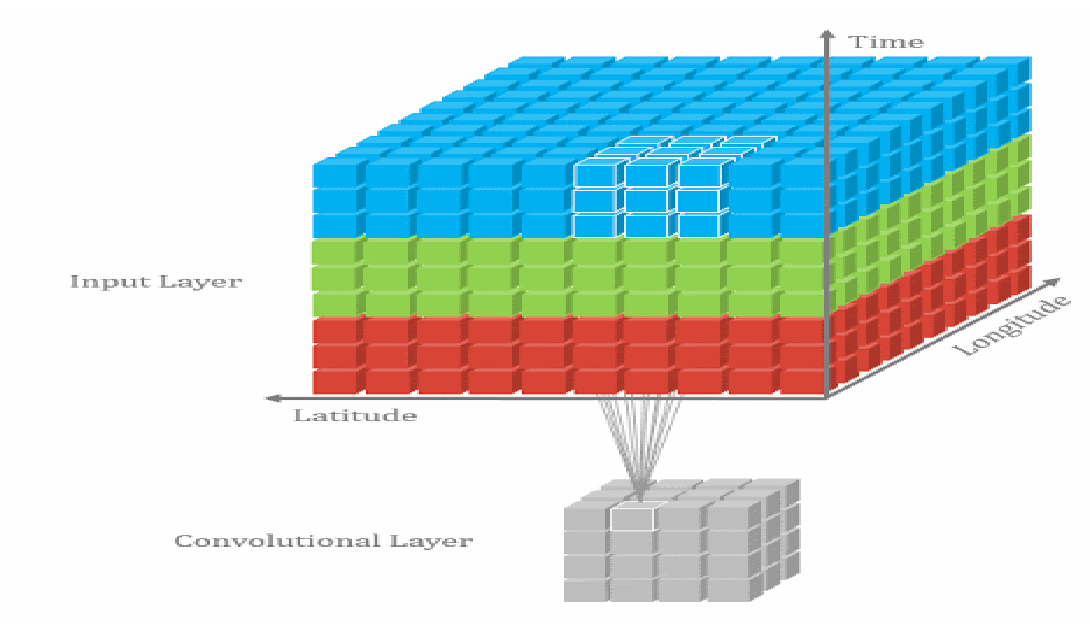
This "patchy" operation is **convolution**



1) Apply a set of weights – a filter – to extract **local features**

2) Use **multiple filters** to extract different features

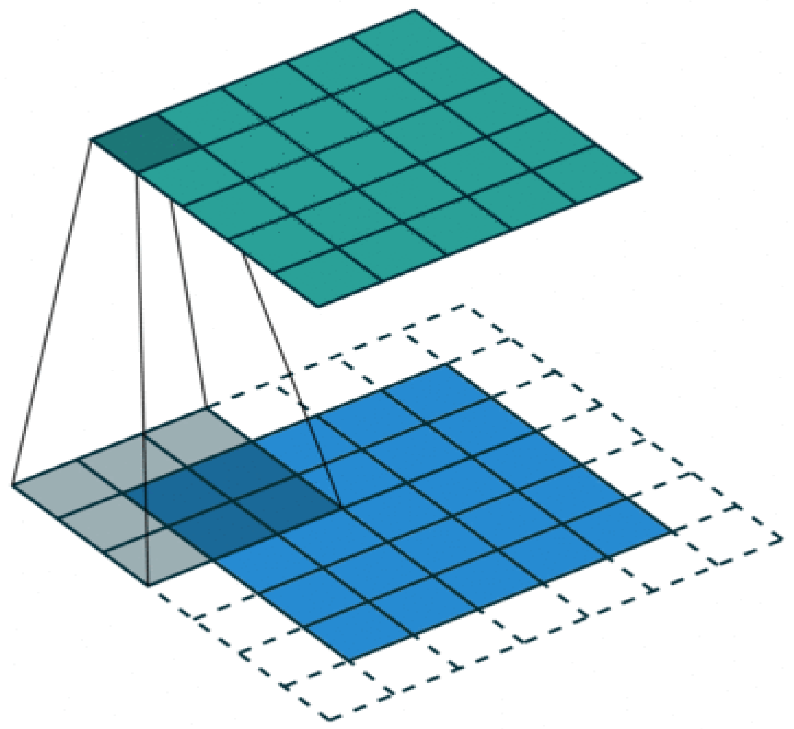
3) **Spatially share** parameters of each filter

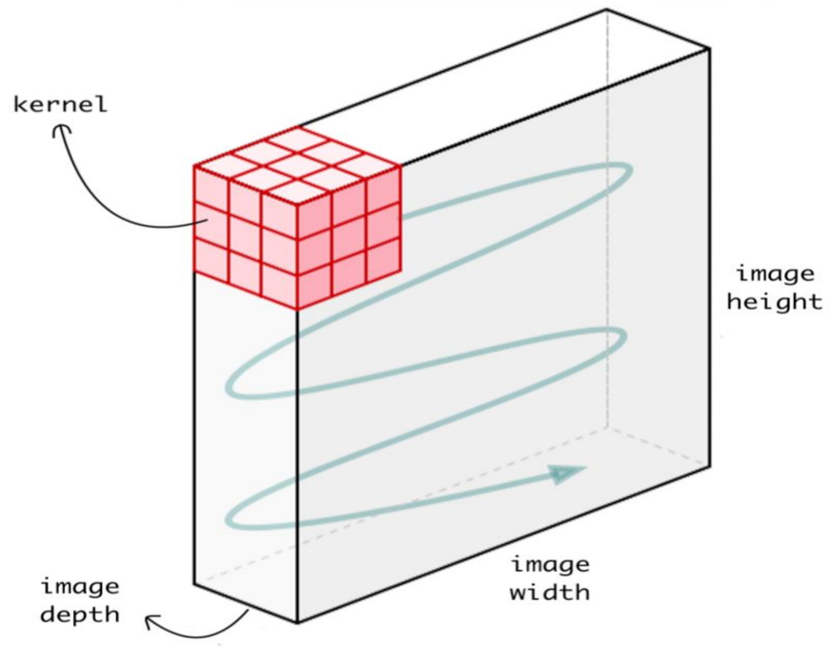


Convolution in Deep Learning

In Deep Learning as relevant to Images and video data, convoluting essentially means that a Kernel(which is a 2D array of weights) sweeps over a block(Kernel size) of individual pixels in an image and *multiplies each pixel value with the kernel weight* at that particular position.This effectively extracts some particular features(which depends upon the values of the kernel) from the Image and is then *pooled(average/max) for further processing.*

A way of visualizing Convolution is shown below:-(Here the stride length is 1 and there is 'same' padding.The 2nd image shows 2D convolution)



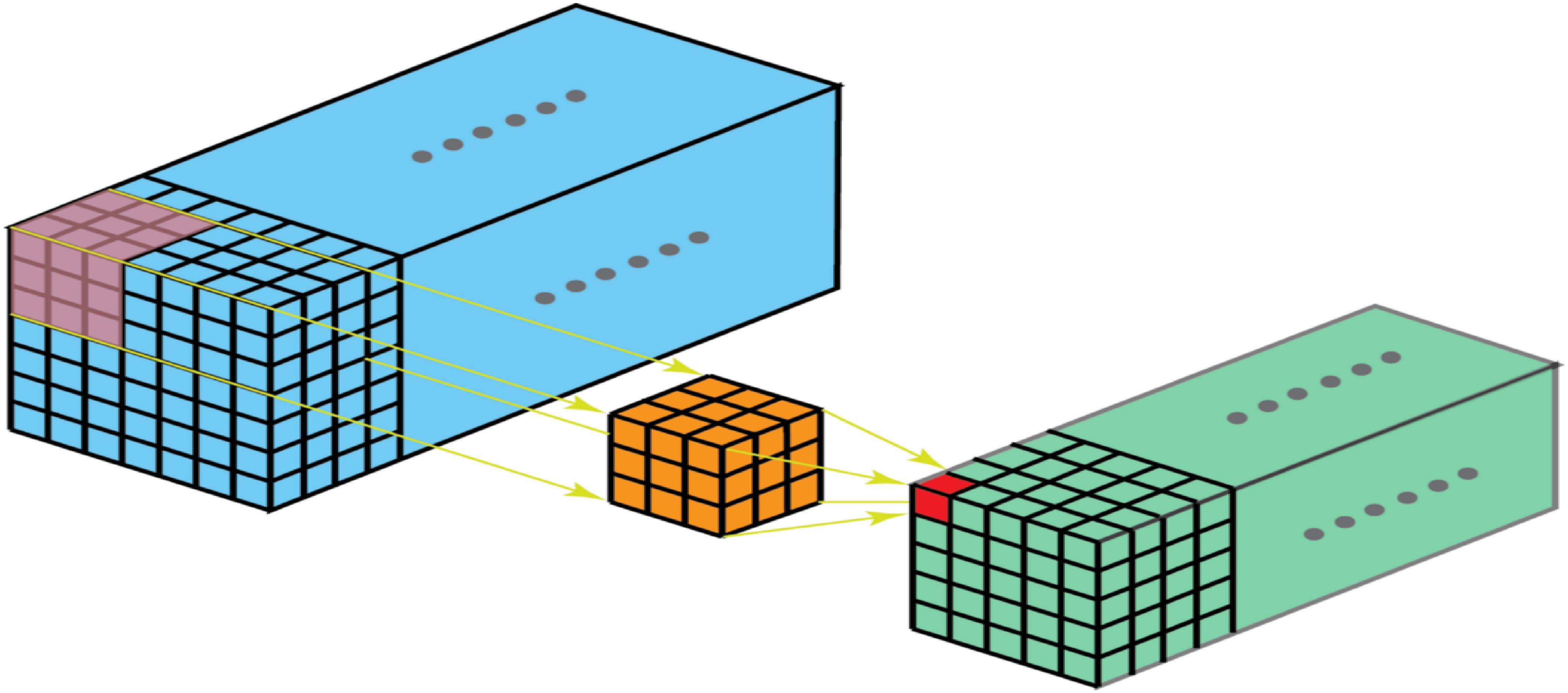


3D Convolution

With Applications and Uses

What is 3D Convolution?

In 3D Convolution, a 3D filter slides over a 3D volume where the *kernel size is less than the channel size*.At each pixel position the kernel value is multiplied with the pixel value and added to give a single number.Since the filter traverses through a 3D volume, the *generated numbers also lie in a 3D space*.This mapping is required for 3D imaging and video data(which is just a collection of related image frames).



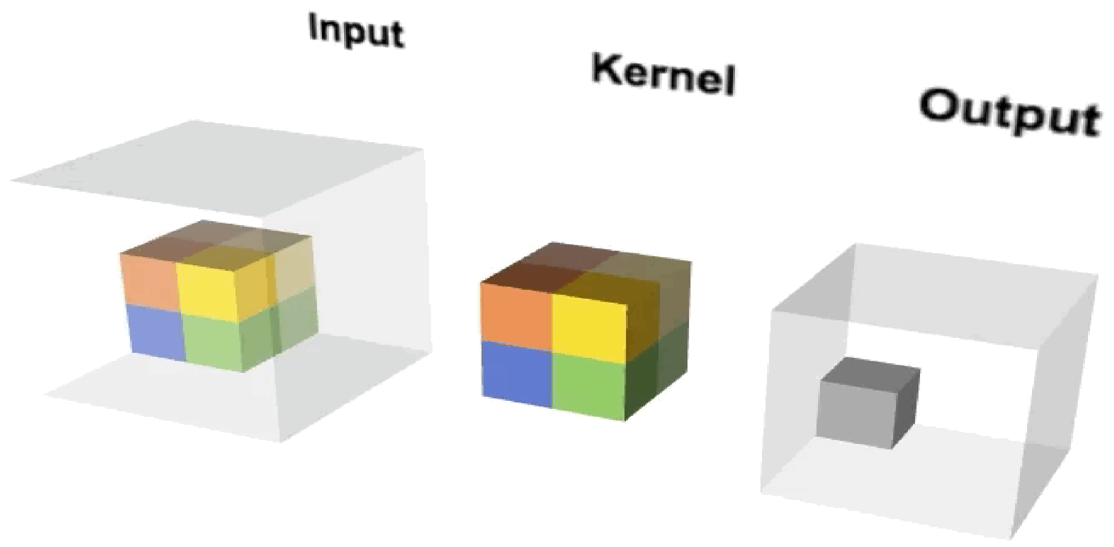
Why 3D convolution?

Basically computer sees an image as 2D matrix of numbers one for each pixel in that image.

In 2D CNNs, convolutions are applied on the 2D feature maps to compute features from the spatial-dimension only.But when it applied to video analysis problems,it is desirable to capture the motion information encoded in multiple contiguous frames.To this end, the 3D convolution occurs in the convolution stages of CNNs to compute features from both spatial and temporal dimensions

CONV2D is called 2 dimensional CNN because the **kernel** slides along 2 dimensions on the data.

The 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together.



Features of 3D Convolution

There are some features which make 3D Convolutions stand out among pre-trained models like ResNet/AlexNet and give better performance than Two stream 2D CNN Models.However they also come with some inherent disadvantages. They are listed as follows:-

PROS:-

- 3D convolutions extracts both spatial and temporal components relating to motion of objects, human actions, human-scene or human-object interaction and appearance of those objects. Thus they are not limited to appearance representation as in ResNets or AlexNet.This makes it really easy for various video related tasks such as action localization and event detection without the need for specific fine-tuning for each task.
- 2D convolution results in convolving with a kernel which results in again a 2D image. The issue in a 2D convolution on a video segment is that the temporal features at this point gets squashed by this operation resulting in a temporally averaged appearance feature map without any motion representation. This can be prevented by using a 3D convolution operation where a kernel can be defined explicitly and the convolution operation takes place between a RGB video segment, giving an output cuboid which preserves the temporal information.
- 3D CNN Models are better at capturing the relationship of dependencies between multiple Image frames and their temporal relationship.Since 2D CNNs(which function based on 2D Convolutional Kernels) take a single slice of input at any given time, they fail to leverage the context from adjacent slices.

CONS:-

- 3D CNNs can be used as low level feature extractors only over multiple short intervals as 3D CNNs fail to capture long term spatio-temporal dependencies. ConvLSTMs can be used to overcome this limitation.
- Training of 3D CNN models require significant computational resources.

Analysis of the 5th paper(Hara et al. 2018)

Highlights of the timeline for the development of 3D convolution and related architectures:-

1.Ji et al, proposes the first 3D Convolutional model where spatiotemporal features could be directly extracted from raw videos.

2.Tran et al, trained a VGG-like CNN which was called the C3D where they showed that a 3*3*3 convolutional kernel achieved the best performance level.

3.Varol et al. showed that expanding the temporal length of inputs for C3D improves recognition performance.

4.Kay et al. showed that using their proposed Kinetics dataset for the training of 3D CNNs significantly improves recognition accuracy.

5.Carreira et al. achieved state-of-the-art performance using inception based 3D CNNs, which was referred to as I3D.

This paper by Hara explores some good practices to be followed while training 3D CNNs.

Experimental setup

Broadly two techniques were used for optimizing the training of the 3D CNNs, they were Data Augmentation and selective Fine tuning of layers. These two were primarily more focused on.ResNet-18 architecture was used throughout the experiment.

For Data Augmentation four configurations were used: Spatial center, Spatial random, multi-scale spatial random and multi-scale spatial corner.

Out of all these, spatial random performed better than the spatial center configuration and multi-scale spatial random achieved the best results overall.Adding spatial variations into training samples is considered to be very important for 3D CNNs.

However the above results indicate a interesting result that the corner cropping strategy while effective for 2D CNNs is not suitable for 3D CNNs and it consistently underperforms than the random spatial position selection strategy. Strategies like RICAP(Random Image cropping and Patching.) also prove that random selection strategies work better in 2D Convolution as well.)

Even among the two random cropping strategies, the performance of *multi-scale spatiotemporal random* was less than that of *multi-scale spatial random*. This indicates that multi-scale temporal cropping decreases the accuracies even though the multi-scale spatial cropping improved recognition accuracies.

Next, the various Fine tuning techniques were tested with each of the above Data augmentation techniques.Below the table(from the Paper) with the performance for each configuration is attached for quick reference.

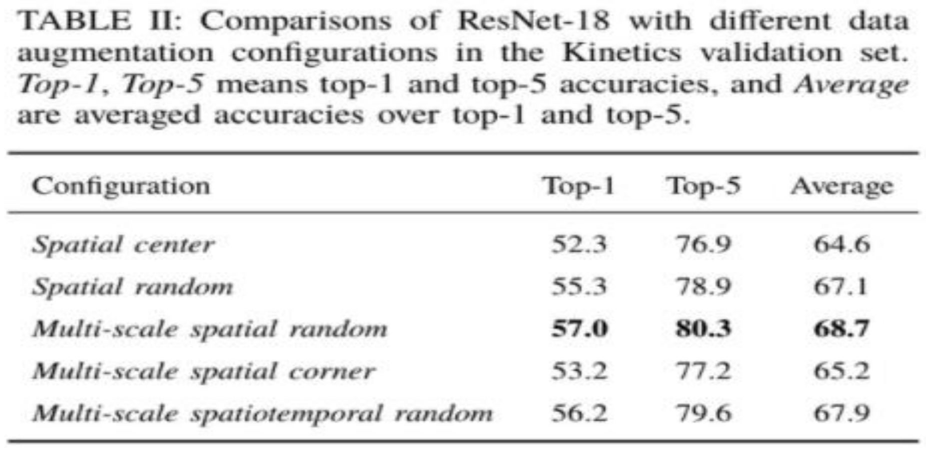


TABLE II: Comparisons of ResNet-18 with different data augmentation configurations in the Kinetics validation set. *Top-1*, *Top-5* means top-1 and top-5 accuracies, and *Average* are averaged accuracies over top-1 and top-5.

Configuration	Top-1	Top-5	Average
Spatial center	52.3	76.9	64.6
Spatial random	55.3	78.9	67.1
Multi-scale spatial random	**57.0**	**80.3**	**68.7**
Multi-scale spatial corner	53.2	77.2	65.2
Multi-scale spatiotemporal random	56.2	79.6	67.9

CONCLUSIONS:-

1.The Spatiotemporal Random cropping Data Augmentation configuration proved to be the most effective after all the trials.

2.Data augmentation by multi-scale spatial cropping increased the accuracies in most cases whereas multi-scale temporal cropping decreased them.

3.A corner cropping strategy proved to be detrimental, which was previously shown as a good method for two-stream 2D CNNs, but which resulted in lowered accuracies for 3D CNNs compared with simple random cropping.

4.Freezing early layers of 3D CNNs improved the performance levels when fine-tuning 3D CNNs on a relatively small dataset.(The larger dataset here being the Kinetics dataset and the UCF-101/HMDB dataset being the smaller ones respectively.)

Analysis of the 6th paper(Hara et al. 2017)

Review of some related works and resources/datasets before this paper:-

1.Action recognition datasets:-

The authors note that UCF-101 and HMDB datasets were the two most successful datasets for action recognition.However, it is extremely difficult to train models on these datasets without overfitting since they are so small.This problem is somewhat solved with the introduction of the recent huge databases such as Sports-1M and YouTube-8M.These datasets though have two drawbacks i.e, their annotations are noisy and video-level labels are not directly related to the targeted activities taking place in the video.

2.Existing Action recognition approaches:-

Two stream CNNs with 2D Convolutional Kernels were the most popular for action recognition.Many works focused on the fusion of the two streams and Feichtenhofer et al. proposed combining two-stream CNNs with ResNets. Kay et al. showed the results of 3D CNNs on their Kinetics dataset are competitive to the results of 2D CNNs pretrained on ImageNet whereas the results of *3D CNNs on the UCF101 and HMDB51 are inferior to the results of the 2D CNNs*. Carreira et al. introduced the inception architecture , which is very deep network (22 layers), to the 3D CNNs and achieved state-of-the-art performance.

Here, the authors focus on developing a model which combines the ResNet architecture with 3D CNNs.

The authors use the ResNet architecture which was the first to introduce shortcut connections that bypass a signal from one layer to the next. The connections pass through the gradient flows of networks from later layers to early layers, and ease the training of very deep networks.

The following Table illustrates the Complete architecture used(taken from the paper):-

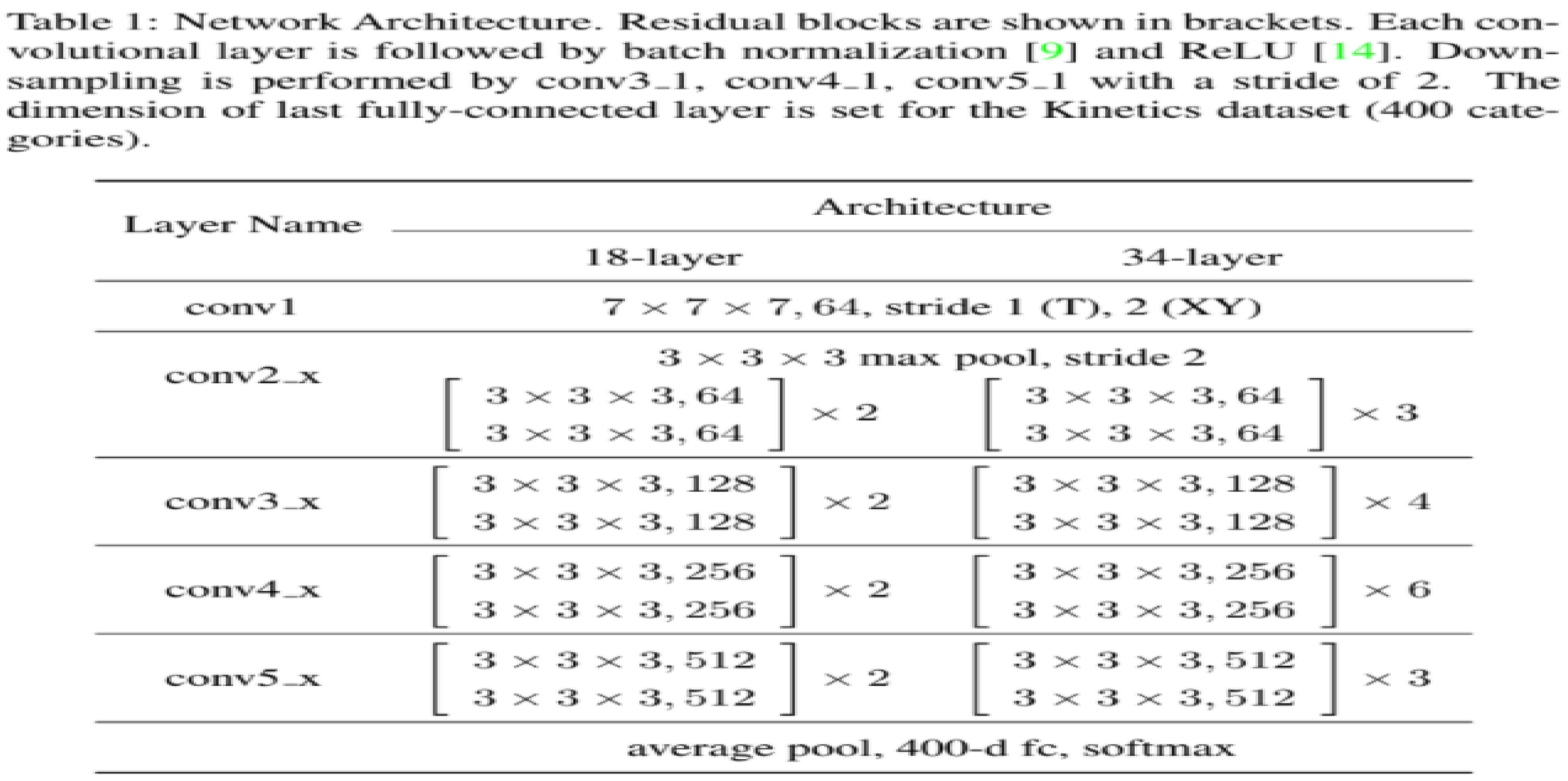


Table 1: Network Architecture. Residual blocks are shown in brackets. Each convolutional layer is followed by batch normalization [9] and ReLU [14]. Downsampling is performed by conv3_1, conv4_1, conv5_1 with a stride of 2. The dimension of last fully-connected layer is set for the Kinetics dataset (400 categories).

Layer Name	Architecture	
	18-layer	34-layer
conv1	$7 \times 7 \times 7, 64$, stride 1 (T), 2 (XY)	
conv2_x	$3 \times 3 \times 3$ max pool, stride 2	
	$\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 3$
	average pool, 400-d fc, softmax	

Implementation and Results

TRAINING:-

Stochastic gradient descent (SGD) with momentum was used to train the network.Temporal positions of each sample were selected by uniform sampling. 16 frame clips were generated around the selected temporal positions.If videos were shorter than 16 frames, they were looped as many times as necessary.For each sample spatial scales are selected from {1,1/(2^(¼)),1/(2^(½)) and half}.(All values from the Paper.)

RECOGNITION:-

We adopt the sliding window manner to generate input clips,(i.e. each video is split into non-overlapped 16 frame clips.) Each clip is cropped around a center position with the maximum scale. We estimate class probabilities of each clip using the trained model, and average them over all clips of a video to recognize actions in videos.

RESULTS:-

ActivityNet v1.3 and Kinetics Datasets were chosen for the paper. The ActivityNet dataset provides samples from 200 human action classes with an average of 137 untrimmed videos per class and 1.41 activity instances per video. The total video length is 849 hours, and the total number of activity instances is 28,108. The dataset is randomly split into three different subsets: training, validation and testing, where 50% is used for training, and 25% for validation and testing.

The Model performs significantly better in-spite of having higher no of parameters.Also, it is pretty competitive to the C3D without being trained on the ImageNet dataset.Carreira et al. used 32 GPUs to train the RGB-I3D whereas here 4 GPUs were used with 256 batch size(GPU:NVIDIA TITAN X).However, computation time for 3D ResNets is very high(3 weeks).

The C3D Architecture

Generic Features for Video Analysis

Introduction

What is C3D?

C3D is a modified version of BVLC Caffe to support 3-Dimensional Convolutional Networks. C3D can be used to train, test, or fine-tune 3D ConvNets efficiently.

C3D STRUCTURE

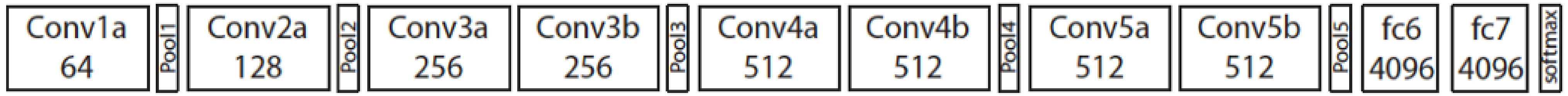
C3D are deep 3-dimensional convolutional neural networks with a homogenous architecture containing 3 x 3 x 3 convolutional kernels followed by 2 x 2 x 2 pooling at each layer. They are trained on a large scale supervised video dataset such as UCF-101 and Sports 1M.

SOME CHARACTERISTICS OF C3D

- The C3D model is given an input video segment of 16 frames (after downsampling to a fixed size which depends on dataset used) and it outputs a 4096-element vector. It gives a compact representation of a video segment which can be further fed to either a temporal network for action localization task , action recognition and action-word mining . Thus, it makes storage and retrieval of videos highly scalable.
- Due to its homogenous architecture with small kernel sizes of 3 x 3 x 3, it can facilitate the optimized implementation of this architecture for embedded platforms. This can be hugely beneficial for smart camera environments where processing of real-time feed and extraction of meta data is critical for real time HAR and others like security survelliance, traffic monitoring etc.

Some more details about C3D

The C3D network architecture consists of 8 convolutional layers, 5 pooling layers and 2 fully connected layers.This is illustrated in the diagram below:-



The first convolution layer of size *1x3x3* is followed by a pooling layer of size *1x2x2.* This is to preserve the temporal information in the first layer and build higher level representation of the temporal information at subsequent layers of the network. Every other convolution layer and pooling layer would have a size of *3x3x3* and *2x2x2* where the strides are 1 and 2 respectively. The fully connected layers have a size of *4096* dimensions with softmax outputs reflecting either 101 classes of UCF-101 dataset or 487 classes of the Sports 1M dataset.

Some other Architectures

1.C3D[2014]

2.Conv3D & Attention[2015]

3. ActionVlad[2017]

4.I3D[2017]

5.T3D[2017]

6.S3D[2017]

References

1.https://www.khanacademy.org/math/differential-equations/laplace-transform/convolution-integral/v/the-convolution-and-the-laplace-transform

2.https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610

3.https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

4.https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

5.https://ai.stackexchange.com/questions/13692/when-should-i-use-3d-convolution#:~:text=Essentially%20its%20the%20same%20as,then%20the%20feature%20map%20depth.

6.https://medium.com/@nair.binum/quick-overview-of-convolutional-3d-features-for-action-and-activity-recognition-c3d-138f96d58d8f

7..https://www.hindawi.com/journals/cin/2017/8348671/

8.https://papers.nips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf

9.https://vlg.cs.dartmouth.edu/c3d/

10.https://arxiv.org/abs/1705.07750

11.https://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review

12.https://paperswithcode.com/task/3d-human-action-recognition

13.https://arxiv.org/abs/1411.4389

14.https://arxiv.org/pdf/1412.0767.pdf

15.https://sci-hub.se/10.1109/tpami.2012.59

16.https://www.youtube.com/watch?v=iaSUYvmCekI

17. https://arxiv.org/pdf/1811.09030.pdf