UNDERSTANDING COMPUTER VISION, 2D CONVOLUTION AND 2D TRANSPOSE CONVOLUTION FUNCTIONS AND THEIR APPLICATIONS

Vision is an art of seeing through which we come to know how an object looks, what features it has. What is its size and many more attributes about the object. Today is an era of artificial intelligence and most of the things are done by computers whether it is object detection or classification. It may be the field of handwriting recognition or features extraction from an image. Machines learn each and everything and perform their tasks very efficiently. But how does a computer see an image and process it. How it extracts features and does classification. In this paper, we describe how an image is stored in a computer and how the images are processed before the training of the model. We mainly focus on the convolution and transpose convolution function and their applications. For clear understanding of vision, we use preprocessed mnist data provided by tensorflow.

How does computer stores images

Computer is a digital device which is mostly comfortable with the numbers. It does not understand colors and features of an object in terms of human language like thin , thick, dark, light etc. It only understands the numbers. So in a computer an image is also stored in a form of matrix numbered from 0 to 255. A black n white image is known as grayscale image having channel one and colored image is stored in RGB format having channel three.

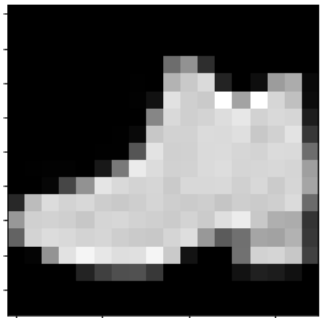
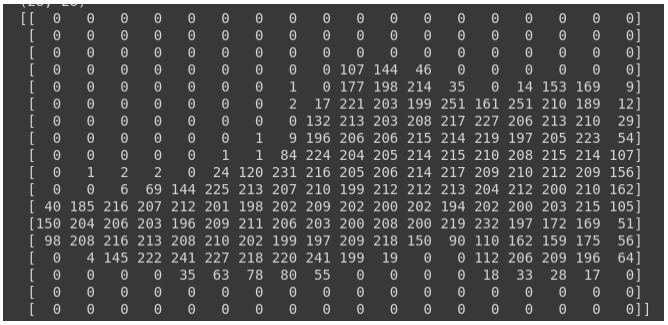


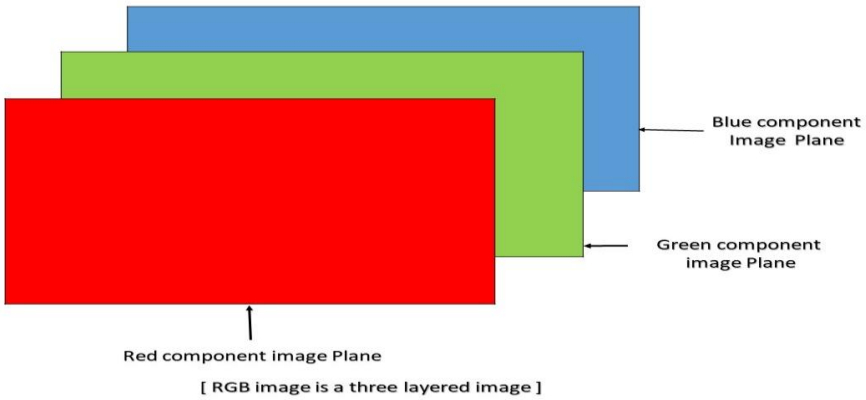
fig (a)



```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 107 144  46   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   1   0 177 198 214  35   0  14 153 169   9   0]
 [  0   0   0   0   0   0   0   0   2  17 221 203 199 251 161 251 210 189  12   0]
 [  0   0   0   0   0   0   0   0   0 132 213 203 208 217 227 206 213 210  29   0]
 [  0   0   0   0   0   0   0   1   9 196 206 206 215 214 219 197 205 223  54   0]
 [  0   0   0   0   0   0   1   1  84 224 204 205 214 215 210 208 215 214 107   0]
 [  0   1   2   2   0  24 120 231 216 205 206 214 217 209 210 212 209 156   0   0]
 [  0   0   6  69 144 225 213 207 210 199 212 212 213 204 212 200 210 162   0   0]
 [ 40 185 216 207 212 201 198 202 209 202 200 202 194 202 200 203 215 105   0   0]
 [150 204 206 203 196 209 211 206 203 200 208 200 219 232 197 172 169  51   0   0]
 [ 98 208 216 213 208 210 202 199 197 209 218 150  90 110 162 159 175  56   0   0]
 [  0   4 145 222 241 227 218 220 241 199  19   0   0 112 206 209 196  64   0   0]
 [  0   0   0   0  35  63  78  80  55   0   0   0   0  18  33  28  17   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

fig(b)

fig(a) is a grayscale image having channel one. You can easily see in fig(b) how a computer reads an image and how it looks for computers as well as humans. A colorful image is a stack of three layers which are known as RGB(red, blue, green) having values between 0 - 255. For example if a pixel is of orange color, then its RGB value will be (255,69,0).

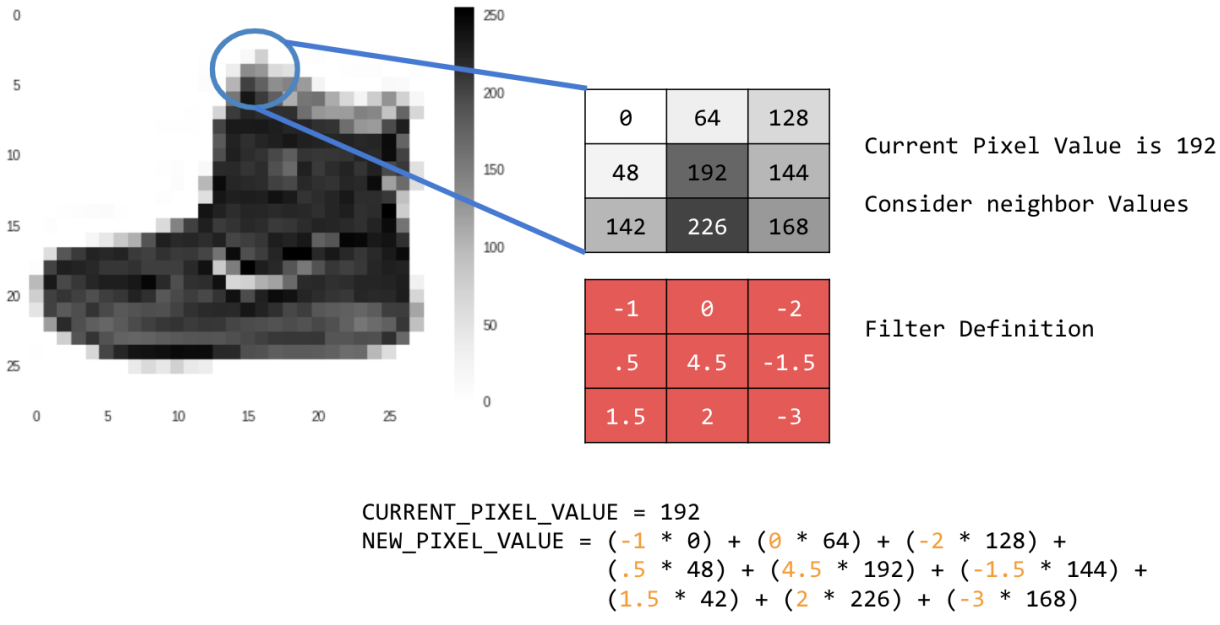


fig(c)

Due to the large size of image, it becomes very cumbersome to train the model even for the simplest task like classifying an image in two classes due to the requirement of high memory and processing power to process each pixel of image. To reduce these resources, before training the model, we process the image and apply some methods to reduce its size or extract its main features. Sometimes we need to generate original dimension images from the extracted features. All these tasks can be fulfilled using convolution and transpose convolution methods which are described below in detail.

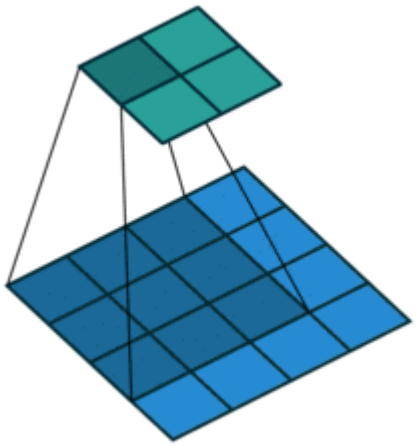
Convolution

A convolution is a method which is used to process the image and compute the new pixel value for the image which shows commonality among them. This new pixel value is computed by multiplying filters with corresponding image pixels and adding them. Following fig(d) shows the procedure.



fig(d)

This process is repeated until the whole image is not covered. By doing so, main features are extracted and size of the image is also reduced. fig(e) shows the convolution function on the image of size 4X4 with filter size 3X3 and the new output image has size of 2X2.



fig(e)

The size of the output image after the convolution operation can be calculated by following formula -

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1.$$

Where

o = size of output image

i = size of input image

p = padding size

k = kernel or filter size

s = stride size

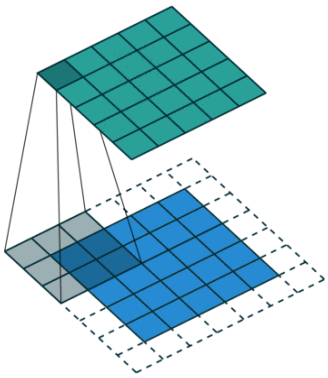
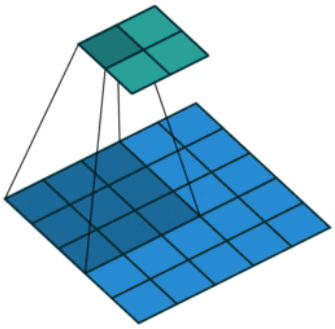


fig (f)



fig(g)

fig(f) shows convolution function with padding = 1 , stride = 1 , kernel = 3 on input of 5X5

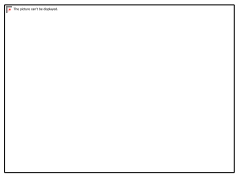
fig(g) shows convolution function with padding = 0 , stride = 2 , kernel = 3 on input of 5X5

Convolution works on the idea that rather than looking at the whole image at once, we can recognize by its main attributes which defines it. For example, a child recognizes an apple from a mango because of its shape and color features. Same procedure is followed by a machine to learn what is apple and what is mango. To recognise fruit, machines will learn its boundary curve, not the whole things like width, height etc. Using filters we can extract the feature which we want as in our case we need to extract the boundary of the fruit in the image.

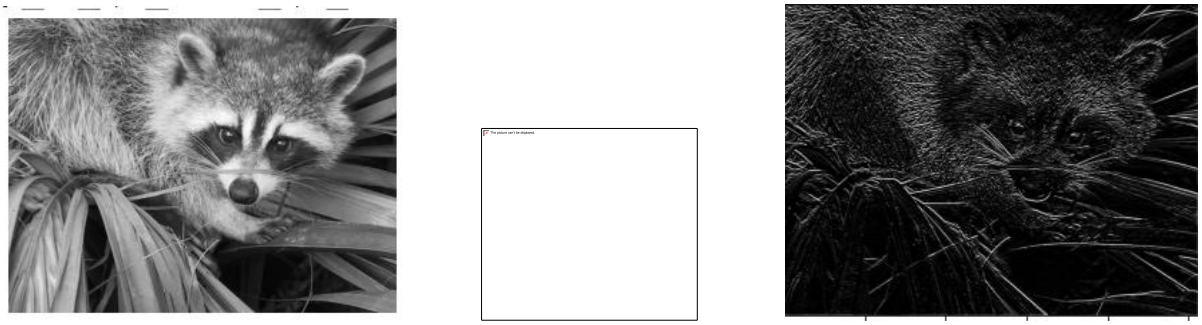
Filter can have any value which is used to extract corresponding features. A vertical filter extracts vertical lines from the image and horizontal filter extracts horizontal lines of the image. Extracted features depend on the value of the filter. fig(h) is the original image which has an animal lying on grass leaves. We convert this image into grayscale and apply convolution function manually to see the effect of the filter on the image. We use horizontal and vertical filters and see their corresponding effects which is shown in fig(j) and fig(k)

fig(h)

fig(i)



fig(j)



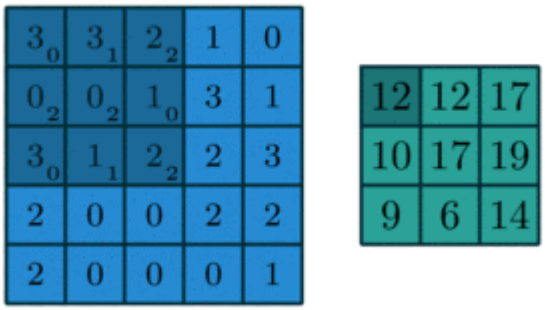
fig(k)

As the difference can be seen very clearly that the grass features are not extracted in fig(k) but the animal is totally extracted. In fig(j) grass is extracted very clearly but the animal's face and hand is not much clear. So, this is clear that we can extract features from the image by using different values of filter.

Types of convolutions

1. **Discrete Convolution**

In this type of convolution, a vector which is received as an input is multiplied with a matrix known as filter or kernel to get a new vector which is known as output. This type of convolution is applicable on images, sound clips or an unordered collection of features etc. In this convolution, multi dimensional data is flattened into a vector and a filter is applied to whole data.



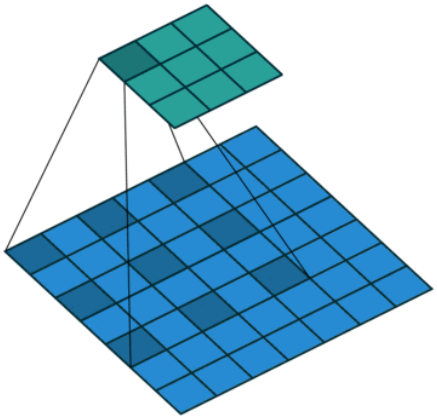
In the above figure, convolution is done on the image using filter[[0,1,2],[2,2,0],[0,1,2]].

2. **Miscellaneous Convolutions**

- **Dilated Convolution**

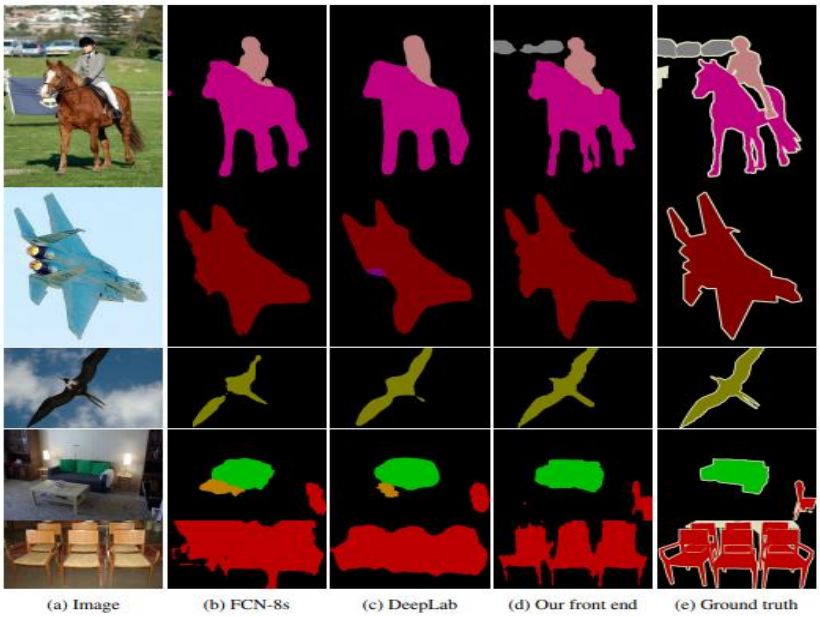
  In this type of convolution, some space is inserted in the kernel and this space depends on hyperparameter d known as dilation rate. If d = 1, then convolution is regular otherwise it has some space between them. In dilation, kernel size is reformed which can be calculated as given below -

  $$\hat{k} = k + (k-1)(d-1).$$



fig(l) Dilation convolution with i = 7, d = 2 , k = 3, s = 1 , p = 0

Dilated convolutions perform better than the regular convolution specially in dense prediction as most of the important pixels are mixed up with each other and don't produce important results by regular convolutions. It helps in aggregating multi-scale contextual information of the image without losing resolution. Systemic dilation supports exponential expansion of the receptive field in the image without losing any resolution and coverage and removing vestigial components. It is more effective than pooling as there is information lost in pooling[1].



(d) is the front end using dilution convolution which is very much similar to ground truth without losing any resolution.

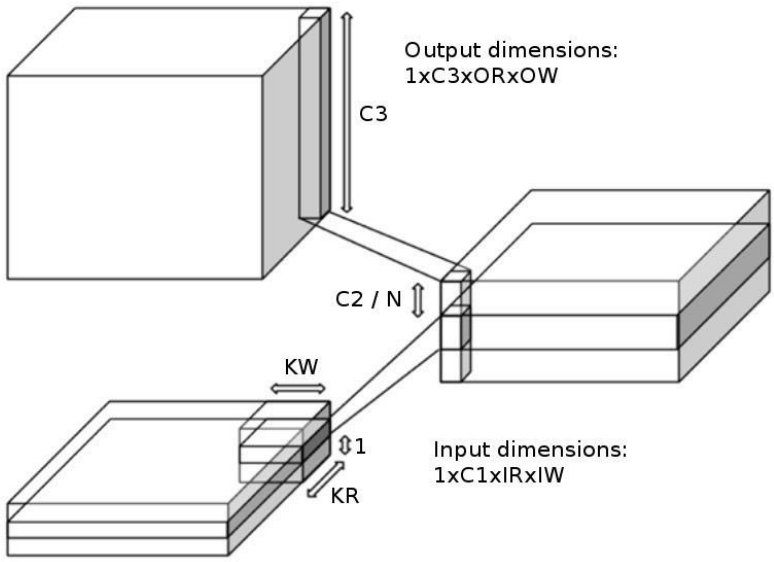
- **Grouped Convolution**

In this type of convolution, the input image and kernel is divided into n distinct groups by their no. of channels no. and convolution is performed on each group independently. The n no. of output of these channels are concatenated to get the overall result. There is a special case, when n is equal to the no. of input channels. This is also known as depth wise convolution or channel wise convolution. It also forms a part of separable convolution.

Aggregated residual transformation of deep neural networks is motivated from grouped convolution in which the architecture is modular and all the outputs are concatenated to get the overall result. This neural network architecture performs very well, even better than the other imagenets neural nets[3].

- **Separable Convolution**

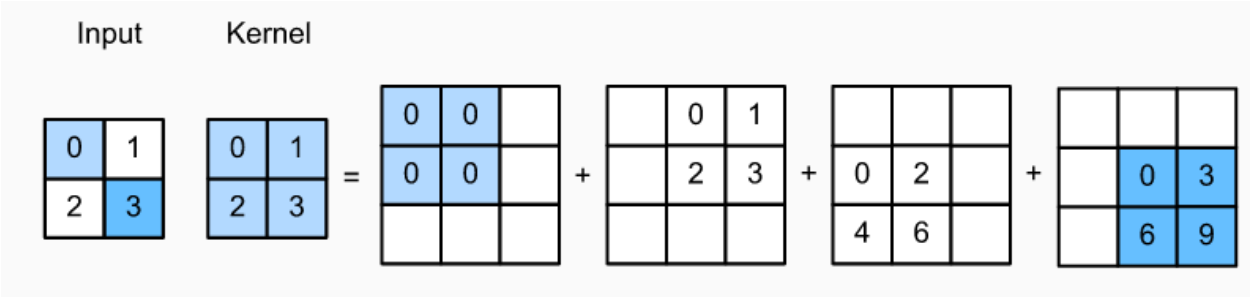
  In separable convolution, convolution is done in two stages - one is depth wise convolution in which no. of groups are equal to the no. of channels. The output of this convolution is the input of of the point wise convolution which is a special case of general convolution of filter size 1X1

- 

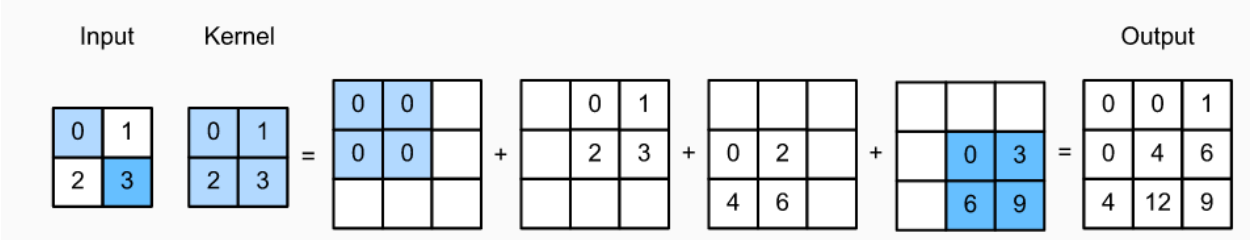
  fig(m)

Transpose Convolution

Transpose convolution is also known as “Fractionally strided convolution” or sometimes preferred as “Deconvolution” which is not appropriate as deconvolution is the removal of convolutional process. Through the transpose convolution, we can get the original dimension of the image from the low dimensional data or latent.



fig(n)



fig(o)

In transpose convolution, a single pixel of input vector is mapped to the size of the filter by multiplying it to the filter or kernel. It is done until the whole input is not taken. After this operation, all the output vectors are summed up to get the dimension of the original image. fig(n) and fig(0) describes the whole procedure.

Formula to calculate the dimension of output image is given below -

$$o' = s(i' - 1) + k - 2p.$$

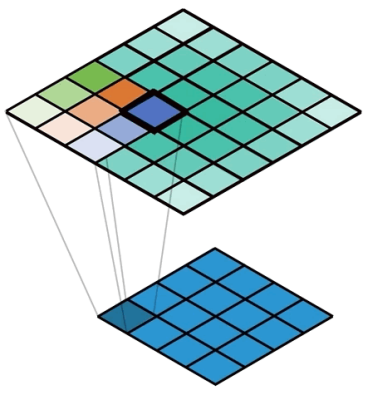
Where o' = size of output image

s = no. of strides

i' = size of input image

k = size of kernel or filter

p = no. of padding



fig(p)

Transpose convolution method is flexible and easy for upsampling of the image but has some disadvantages.The disadvantage of generating images from transpose convolution is checkerboard artifacts[5] in the output image. And the second disadvantage is that we cannot apply this method until we know the weights of the kernel. This problem is not in classical upsampling methods like bicubic, bi-linear interpolation[6] and nearest neighbor interpolation[6].
To reduce the effect of checkerboard artifacts and to make high resolution images, there should be strides in the kernel to avoid overlap issues in image.

Application of Convolution and Transpose Convolution

Convolution and Transpose convolution has a wide range of applications and is the most efficient and easy way for image processing.

- Transpose convolution is used for up-sampling of images and convolution is used for down sampling.
- It is used for the super resolution of the image[7].
- Autoencoder convolution is used as an encoding technique and transpose convolution is used as a decoding technique.
- Transpose Convolution and Convolution methods are also used for semantic segmentation of images[10].
- Convolution is very useful in creating a low dimensional latent of the image by extracting main features.
- Feature extraction using convolution is very useful in object detection, classification[9] and recognition.
- Convolution can be used with the generators for generating images using unsupervised learning[8].

Conclusion

In this paper, we discuss how computers store an image. What is convolution and transpose convolution and in what field they are used. We also describe the mathematics behind convolution and transpose convolution methods and their different types with their advantages and disadvantages.

References

[1] Multi-scale context aggregation by dilated convolution by Fisher Yu , Vladlen Koltun published in ICLR 2016

[2] A guide to convolution arithmetic for deep learning by Vincent Dumoulin, Francesco Visin published in arXiv in 2018

[3] Aggregated Residual Transformations for Deep Neural Networks by Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, Kaiming He, UC San Diego, Facebook AI Research published in arXiv in 2017
[4]Mathematics in Transposed convolution explained | Vignesh Kumar Korakki , medium.com https://medium.com/@kvkumar1993/mathematics-in-transposed-convolution-explained-vignesh-kumar-korakki-bf133c74958
[5] Deconvolution and Checkerboard Artifacts by Augustus Odena, Chris, Vincent Domoulin published in 2017
[6] Transposed Convolution Demystified by Divyanshu Mishra https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba#:~:text=For%20instance%2C%20a%20stride%20of,pass%20of%20a%20normal%20convolution.
[7]SRFeat: Single Image Super-Resolution with Feature Discrimination by Seong-Jin Park, Hyeongseok Son, Sunghyun Cho, Ki-Sang Hong, Seungyong Lee published in ECCV 2018
[8]Unsupervised representation learning with deep convolutional generative adversarial network by Alec Radford & Luke Metz, Soumith Chintala published in arXiv 2016.
[9]ImageNet Classification with Deep Convolutional Neural Networks by Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton published in NIPS 2012
[10] Semantic image segmentation with deep convolutional nets and fully connected CRFs by Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille published in ICLR 2015