STAT – 517

ABHISHEK KUMAR THAKUR

MATERIALS SCIENCE & ENGINEERING

RECENT ADVANCES IN CONVOLUTIONAL NEURAL NETWORKS

ABSTRACT

Deep learning is used in a number of technologies. It has good performance in object recognition, speech recognition and natural language processing. Among different types of deep neural networks, Convolutional Neural Network (CNN) is used much extensively. In early days, due to small number of training data and computer power, it was very difficult to train neural networks. But now a day, we can have large amount of training data as well as computing power and that is why many researchers have achieved state of the art. In this paper, a broad survey is carried out in advances in CNN's and its application in computer vision.

INTRODUCTION

CNN is first introduced by Yann LeCun et.al in 1998 and improved by B. B. LeCun in 1990. They developed a multi-layer neyral network, LeNet – 5 which can classify handwritten digits. LeNet – 5 can be trained using backpropagation. LeNet – 5 did not performed well in complex problems i.e large scale image and video classification due to the lack of computing power. Since 2006, many neural network models came to overcome these difficulties. Krizhevsky et.al proposed a classic CNN (AlexNet) which shows improvements over image classification and has a deeper structure than LeNet – 5. AlexNet has 8 layers which consist of 5 convolutional layers and 3 fully connected layers. Rectified Linear Unit (ReLU) is used as a non-linear activation function. Several other developments were made to improve AlexNet and thus led to the discovery of ZFNet, VGGNet, GoogleNet etc.

BASIC COMPONENTS OF CNN

A CNN consists of 3 types of layers:

- Convolution layer (to learn feature representation of inputs)
- Pooling layer (to achieve spatial invariance by decreasing resolution)
- Fully connected layer (to perform high level reasoning)

By stacking convolutional and pooling layers, we could extract more abstract feature representations. In a fully connected layer, all neurons in the previous layer is connected to every single neuron in the current layer. The output of the last fully connected layer is fed to the output layer. For classification tasks, softmax regression and support vector machines are used.

IMPROVEMENTS ON CNN

- CONVOLUTIONAL LAYER
  Convolutional filter in a CNN is a Generalized Linear Model (GLM). Two works are explained below which aims to enhance representation ability.
  1. NETWORK IN NETWORK (NIN)
     This is proposed by Lin et.al in 2013. This replaces the linear filter of the convolutional layer by a micro network which is capable of extracting more abstract information. The overall structure of NIN is the stacking of such micro-networks. To see the difference between NIN and a convolutional layer, we see the feature map in both cases.
     The feature map of a convolutional layer is given by:
     $$f_{i,j,k} = \max(w_k x_{i,j}, 0)$$
     where 'i' and 'j' are the pixel indices of feature map and 'k' is the chanel index of the feature map.
     The feature map of NIN is given by:
     $$f^1_{i,j,k_1} = \max(w^1_{k_1} x_{i,j} + b_{k_1}, 0)$$
     $$f^n_{i,j,k_n} = \max(w^n_{k_n} f^{n-1}_{i,j} + b_{k_n}, 0)$$
     where 'n' is the number of layers in NIN.
  2. INCEPTION MODULE
     This is introduced by Szegedy et.al in 2014. It is a logical culmination of NIN which uses a variable filter size to compare different visual patterns of different sizes and approximates the optimal structure. With this module, the network parameters are dramatically reduced 5 million as compared to AlexNet (60 million) and ZFNet (75 million).
- POOLING LAYER
  It is important in CNN because it lowers the computational burden by reducing the number of connections between convolutional layers. Some recent pooling methods used in CNN are:
  A. Lp – Pooling
     It is a biologically inspired pooling process modeled on complex cells which provides better generalization than max pooling.

$$L_p = \left(\sum_{i=1}^{N} |x_{Ii}|^p \right)^{\frac{1}{p}} = \begin{Bmatrix} Avg.\ pooling, when\ p = 1 \\ L_2\ pooling, when\ p = 2 \\ Max\ pooling, when\ p = \infty \end{Bmatrix}$$

where x_{Ii} = Finite input nodes.

B. Mixed pooling

It is a combination of Max pooling and Average pooling.

$$Mixed\ pooling = \lambda . \max_{(p,q) \in R_{i,j}} . x_{kpq} + (1 - \lambda) . \frac{1}{|R_{ij}|} . \sum_{(p,q) \in R_{i,j}} x_{kpq}$$

where, x_{kpq} = output of pooling operator for (i,j) position in kth – map.

λ either 0 (max pooling) or 1 (average pooling).

R_{ij} = Local neighborhood of (i, j).

C. Stochastic pooling

It ensures that the non-maximal activations of feature maps are also possible to be utilized. It has advantages of max-pooling and can avoid overfitting due to stochastic component.

D. Spectral pooling

It performs dimensionality reduction by cropping input in frequency domain.

$$x \in R^{M \times N} \rightarrow H \times W \rightarrow Discrete Fourier Transform(DFT) \rightarrow Crops\ H \times W \rightarrow Inverse DFT \rightarrow Output$$

E. Spatial pyramid pooling

It can generate a fixed length representation regardless of input size. It pools input feature map in local spatial bins with sizes proportional to the image size which results in a fixed number of bins.

F. Multi-scale orderless pooling

It is used to improve the invariance of CNN's without degrading their discrimination power. The activities of local patches are aggregated by VLAD encoding which aims to capture more local, fine-grained details of the image as well as enhancing variance.

- ACTIVATION FUNCTIONS

A proper activation function improves performance of a CNN. Various improvements in the activation functions are discussed below:

- ReLU

  This is acronym for Rectified Linear Unit. It is a non standard activation function.

  ReLU = $y_i = \max(0, Z_i)$

  where Z_i = Input of the ith channel.

  It is a piecewise linear function which prunes the negative part to 0 and retains only the positive part. A simple 'max' function of ReLU allows it to compute much faster than the sigmoid function.

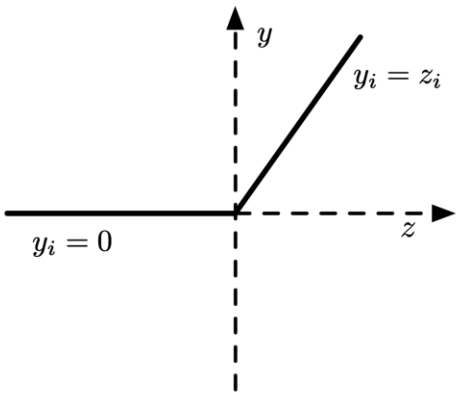


Figure 1: ReLU

- Leaky ReLU

  Compared to ReLU, leaky ReLU compresses the negative part rather than mapping it to constant zero.

  $$y_i = \begin{cases} z_i, z_i \geq 0 \\ az_i, z_i < 0 \end{cases}$$

  where, a = predefined parameter in range 0 to 1.

- Parametric ReLU (PReLU)

  This adaptively learns the parameters of the rectifiers in order to improve accuracy.

  $$y_i = \begin{cases} z_i, z_i \geq 0 \\ a_i z_i, z_i < 0 \end{cases}$$

  where, a_i = learned parameter of i^{th} channel.

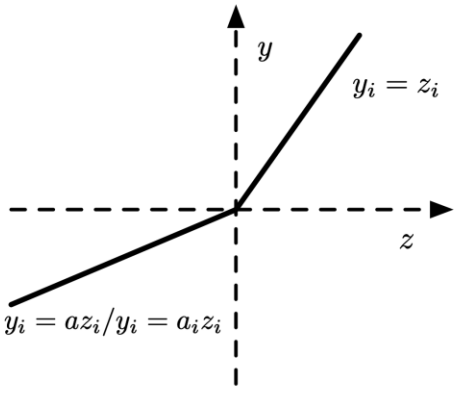


Figure 2: LReLU/PReLU

- Randomized ReLU (RReLU)

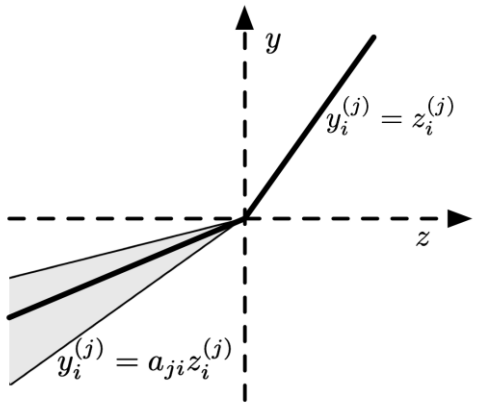
  The parameters of the negative part are randomly sampled from a uniform distribution in learning and then fixed in testing.

  $$y_i^{(j)} = \begin{cases} z_i^{(j)}, z_i^{(j)} \geq 0 \\ a_{ji} z_i^{(j)}, z_i^{(j)} < 0 \end{cases}$$

where, $z_i^{(j)}$ is the i^{th} channel in the j^{th} example,

$a_i^{(j)}$ is the corresponding sampled parameter,

$y_i^{(j)}$ is the corresponding output.



Fugure 3: RReLU

- Exponential Linear Unit (ELU)

  It enables faster learning of deep networks and leads to higher classification accuracy. It employs a saturation function to handle the negative parts.

$$y_i = \begin{cases} z_i, z_i \geq 0 \\ a[\exp(z_i) - 1], z_i < 0 \end{cases}$$

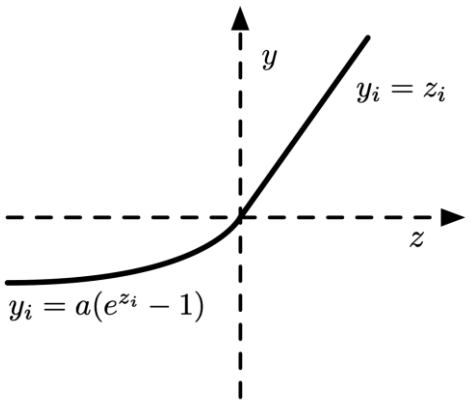


Figure 4: ELU

- Maxout

  It is an alternative non-linear function that takes maximum response across multiple channels at each spatial position.

$$y = \max_{i \in [1,k]} z_i$$

  where, z_i = i^{th} channel of the feature map.

- Probout

  It is a probabilistic variant of ‘maxout’. In this the probability of each linear unit is defined.

$$p_i = \frac{e^{\lambda z_i}}{\sum_{j=1}^{k} e^{\lambda z_j}}$$

where, λ is the hyperparameter to control variance in distribution.

- LOSS FUNCTION

  It is very important to choose correct loss function in a CNN for a specific work. There are several advancements in the Loss function and some of them are discussed below:

  i. Softmax Loss

  It is a combination of multinomial logistic loss and softmax. For a given input, softmax loss is given by:

  $$L_{soft\max} = -\frac{1}{N}\left[\sum_{i=1}^{N}\sum_{j=0}^{k-1} 1\{y^{(i)} = j\}.\log p_j^i\right]$$

  ii. Hinge Loss

  It is used to train large margin classifiers such as SVM.

  $$L_{Hinge} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=0}^{k-1}\left[\max\left(0, 1-\delta(\ln, k)w^T.x_n\right)\right]^p$$

  where, $\delta(\ln, k) = \begin{Bmatrix} 1, if\ \ln = k \\ -1, if\ \ln \neq k \end{Bmatrix}$

  when p = 1 , L1 loss and when p = 2, L2 loss.

  iii. Contrastive Loss

  It is used to train Siamese network. The loss is defined for every single layer and backpropagations are also performed at the same time.

  $$L_l = (y)d_l + (1-y)\max(m - d_l, 0)$$

  where, $d_l = \left\| z_\alpha^l - z_\beta^l \right\|_2^2 = similarity between z_\alpha^l \ and \ z_\beta^l$ and m is the margin parameter.

- REGULARIZATION

  Overfitting can be reduced by regularization.

  - Dropout

    It is very effective in reducing overfitting. It is generally preferred when the training data is small. It also prevents the network from becoming too dependent on any one neuron and force the network to be accurate even in the absence of certain information.

  - DropConnect

    It chooses the weights of a fully connected layer with a probability '*p*'. The output of dropconnect is given by:

    r = a((m * w)v)

    where, $m_{ij} \sim$ Bernoulli(p)

- OPTIMIZATION
    - Weights Initialization
      To achieve fast convergency, proper initialization of weight is required. We set randomly the weights of the input according to Gaussian distribution.
    - Stochastic Gradient Descent (SGD)
      Parallelized SGD method is used for large scale machine learning. Downpour SGD over cluster of computers [Jaffery Dean et.al.]. Multiple GPU's are used to asynchronously calculate gradients & update global parameters. This made the process 3.2 times faster.
    - Batch Normalization
      This is proposed by Serjey Ioffe & Christian Szeged. This is aimed to accelerate the entire training process. They do normalization on the training dataset that fixes mean & variances of input layers.

FAST PROCESSING OF CNN's

a) Fast Fourier Transforms (FFT's)
   The fourier transformation of filters can be reused in a mini-batch operation. The fourier transforms of output gradients can be reused when backpropagating to both filters and input images. It is costly because using FFT to perform convolutions needs additional memory to store feature maps in fourier domain.
b) Matrix Factorization
   Given an (m × n) matrix 'A' of rank 'r', there exists a factorization A = B × C, where 'B' is an (m × r) full column rank matrix & 'C' is an (r × n) full row rank matrix. Thus we can replace A by B × C. This results in 30 – 50% speedup in training process & little loss in accuracy.
c) Vector Quantization
   This is used to compress dense layers to make CNN models smaller. Generally a large set is mapped into a smaller set.

APPLICATION OF CNN's

a. Image Classification
   CNN has been used for image classification for a long time and has better classification accuracy because of joint feature learning and classifier learning. They uses hierarchy of classes for sharing information among related classes in order to improve performance with very few training examples. CNN builds a tree-like structure to learn fine-grained features for subcategory recognition.
b. Object Tracking
   It plays an important role in computer vision application. First of all, Fan et. al used CNN for object tracking purposes. For this propose, CNN is designed as a tracker with shift – variant architecture and trained inclemently during tracking with new examples online.

c. Pose Estimation

   Many recent works pay more attention to learn multiple levels of representations and abstraction for human body pose estimation task. Deep pose is the first application of CNN in pose estimation. Arjun et.al present a CNN based end to end learning approach for full human body pose estimation. In addition to still pose estimation with CNN, it is also applied for pose estimation in videos.

d. Text Detection & Recognition

   It is widely used in CNN (OCR is major focus). Earlier constrained texts recognition is done but now due to computer technologies, unconstrained texts recognition are also taken into account.

   - Text detection & localization without recognition.
   - Text recognition on cropped text images.
   - End-to-End text spotting.

e. Visual Saliency Discretion

   This is the technique to locate important regions in imagery. It is a challenging research topic. The local context is handled by a CNN model which assigns a local saliency to each pixel, while global context is handled by a fully deep connected feedforward network.

f. Action Recognition

   This is broadly classified into 2 groups: Action analysis in still images & Action analysis in videos. Both the groups uses CNN for recognition purpose.

g. Scene Labelling

   The goal is to relate one semantic class (Glass, Road, Water etc.) to each pixel. CNN is able to learn strong features and classifiers to discriminate the local vision subtleties.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a backpropagation network,” in Advances in neural information processing systems. Citeseer, 1990.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[4] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in ECCV, 2014.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” CoRR, vol. abs/1409.4842, 2014.

[6] M. D. Zeiler and R. Fergus, “Stochastic pooling for regularization of deep convolutional neural networks,” CoRR, vol. abs/1301.3557, 2013.

[7] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv preprint arXiv:1502.03167, 2015.

[8] N. Srivastava and R. R. Salakhutdinov, “Discriminative transfer learning with tree-based priors,” in Advances in Neural Information Processing Systems, 2013, pp. 2094–2102.

[9] Z. Yan, V. Jagadeesh, D. DeCoste, W. Di, and R. Piramuthu, “Hd-cnn: Hierarchical deep convolutional neural network for image classification,” arXiv preprint arXiv:1410.0736, 2014.

[10] J. Fan, W. Xu, Y. Wu, and Y. Gong, “Human tracking using convolutional neural networks,” Neural Networks, IEEE Transactions on, vol. 21, no. 10, pp. 1610–1623, 2010.