Deep Learning in HEP

Miroslav Kubů

FNSPE CTU

21.6.2018

Overview

- Artificial Neuron
- Neural Networks
- Convolutional Neural Networks (CNN)
- CNN for Digit Recognition
- CNN in HEP

Artificial Neuron

Artificial neuron is a set $(f, \vec{w}, b)$, where

- $f : \mathbb{R} \to \mathbb{R}$ is called activation function,
- $\vec{w} \in \mathbb{R}^n$ vector of weights,
- $b \in \mathbb{R}$ bias.

Output ... $y = f(\sum_j w_j x_j + b)$

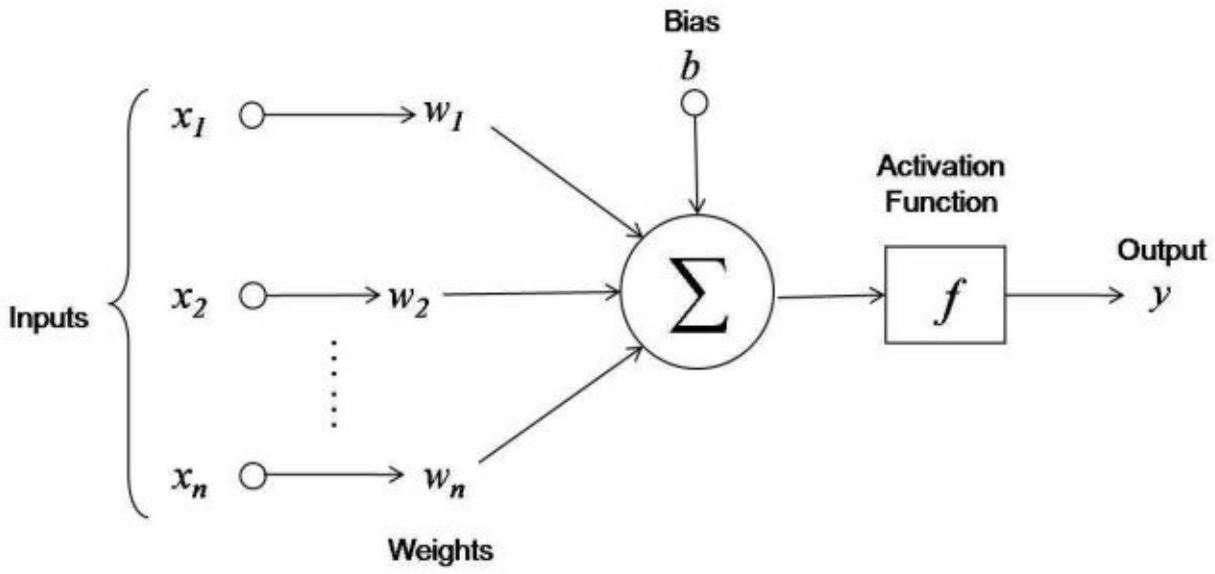

Figure 1: Model of artificial neuron.

Artificial Neural Networks (ANN)

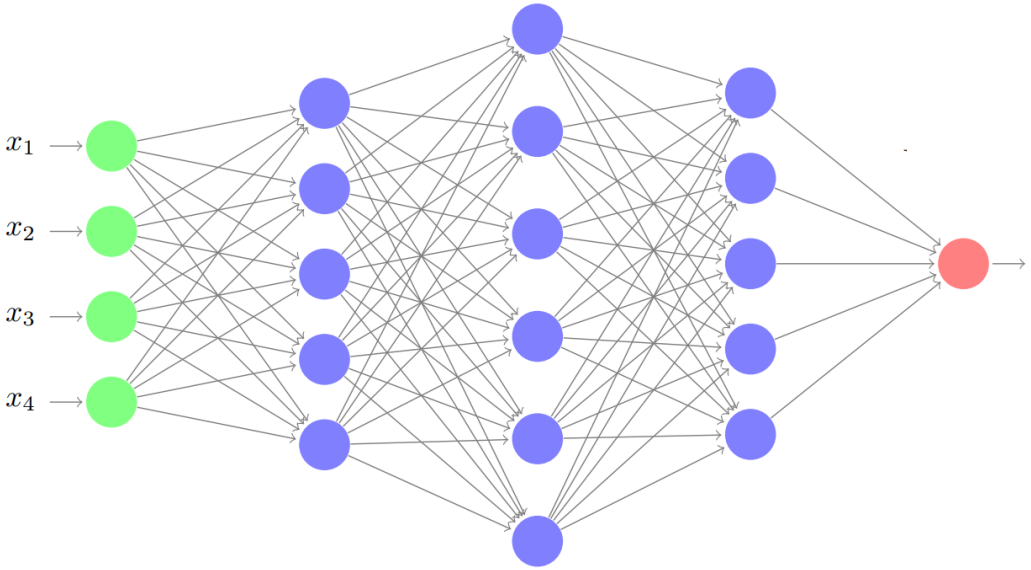

Figure 2: Artificial neural network.

ANN for Image Recognition

- Flatten the image and feed it through ANN as a vector
- High dimensions $\Rightarrow$ huge number of parameters
- No spatial structures recognition

Convolutional Neural Networks (CNN)

Core concepts

- Local receptive fields
- Weight sharing
- Local structure recognition

Architecture

- Convolutional Layer
- Pooling
- Fully Connected Layer

MNIST Dataset

- Set of 28x28 examples of handwritten digits
- 60 000 train examples, 10 000 test examples
- Pixels corresponding to grayscale levels

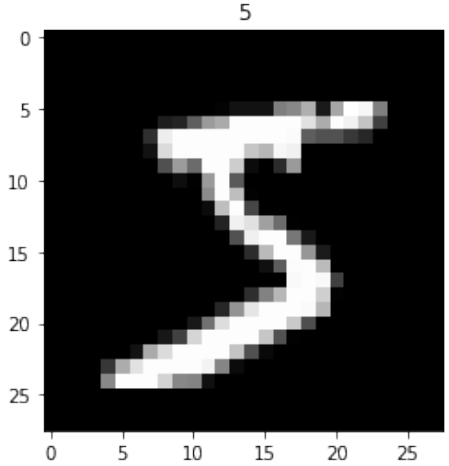

Figure 3: Example of MNIST Dataset.

Convolutional Layer

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \quad (1)$$

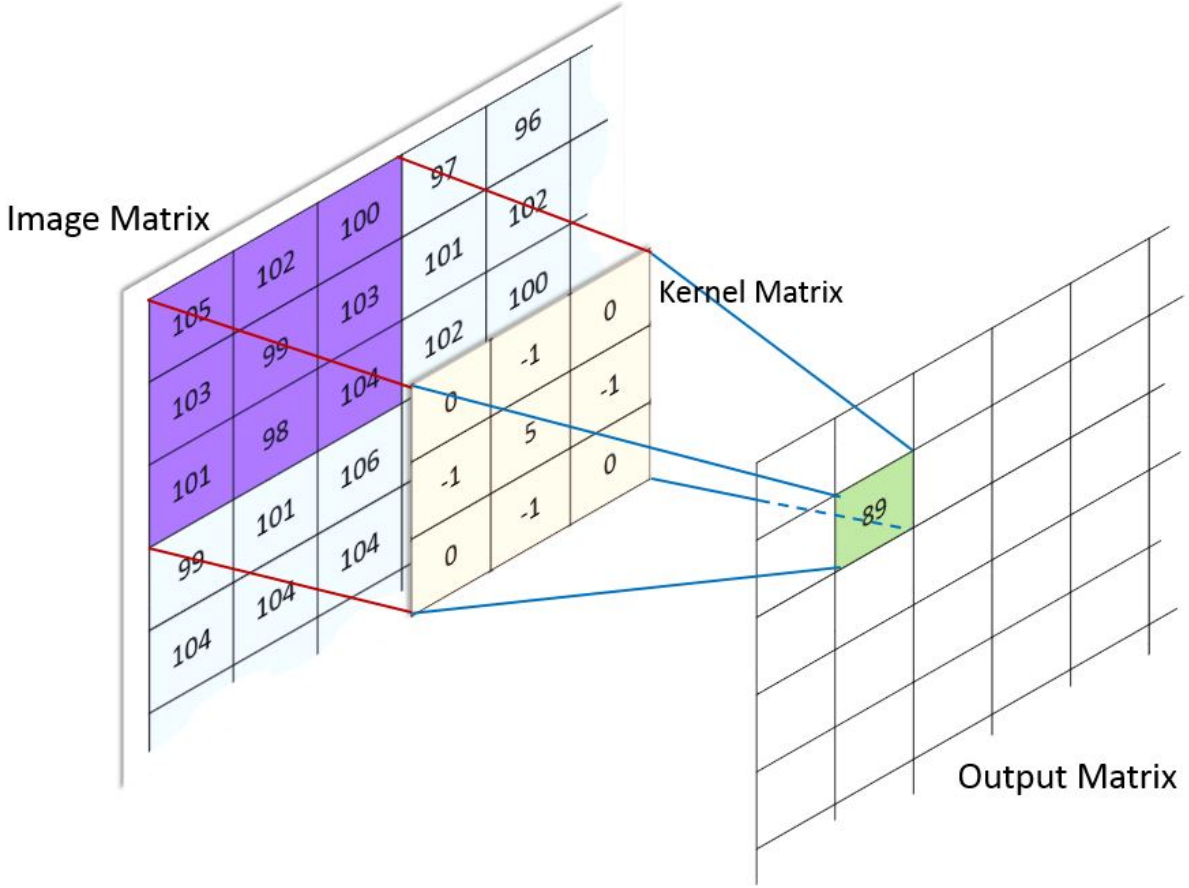

Figure 4: Convolution by 3x3 convolutional filter.

Convolution Layer

Forward pass

$$\xi^l_{x,y} = \sum_{k,h} w^l_{k,h} a^{l-1}_{k+x,h+y}, \tag{2}$$

$$a^l_j = f(\xi^l_j) \tag{3}$$

Activation functions

- ReLU (Rectified Linear Unit)

$$f(\xi) = \begin{cases} \xi & \xi \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

- Softmax function

$$f_j(\vec{\xi}) = \frac{e^{\xi_j}}{\sum_{k=1}^{n} e^{\xi_k}}, \quad j \in \{1, 2, \ldots, n\}. \tag{5}$$

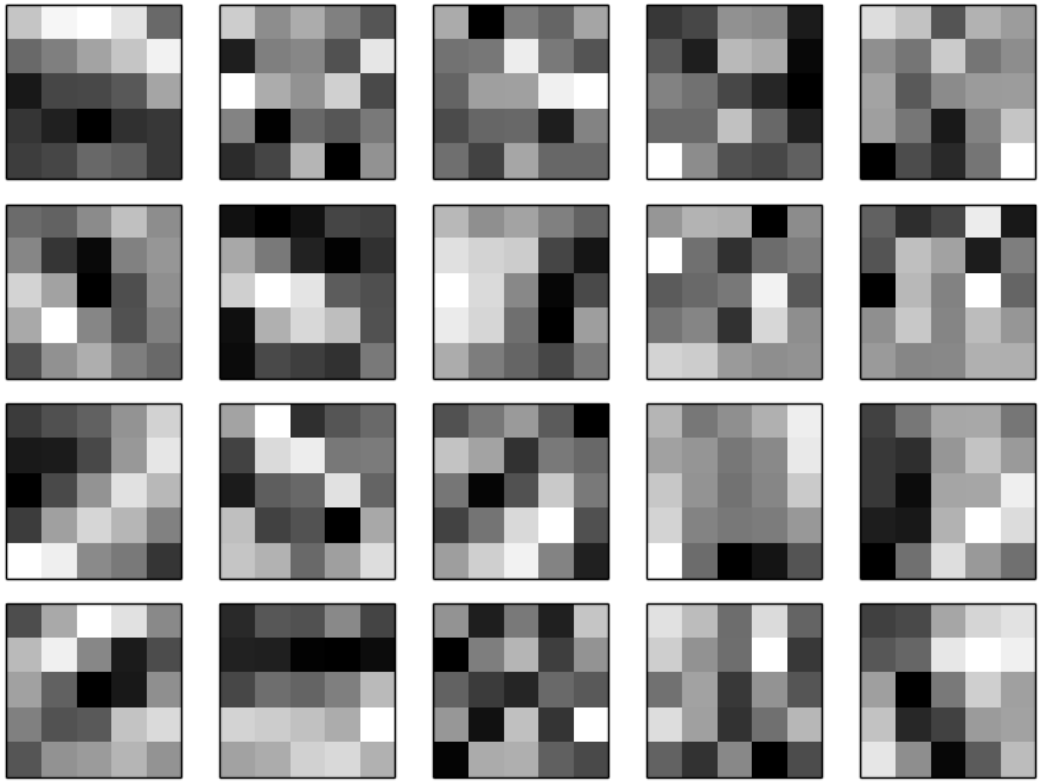

Figure 5: Learned features of 5x5 filter on MNIST Dataset.

Pooling

- Restraining the dimension
- Transition invariance

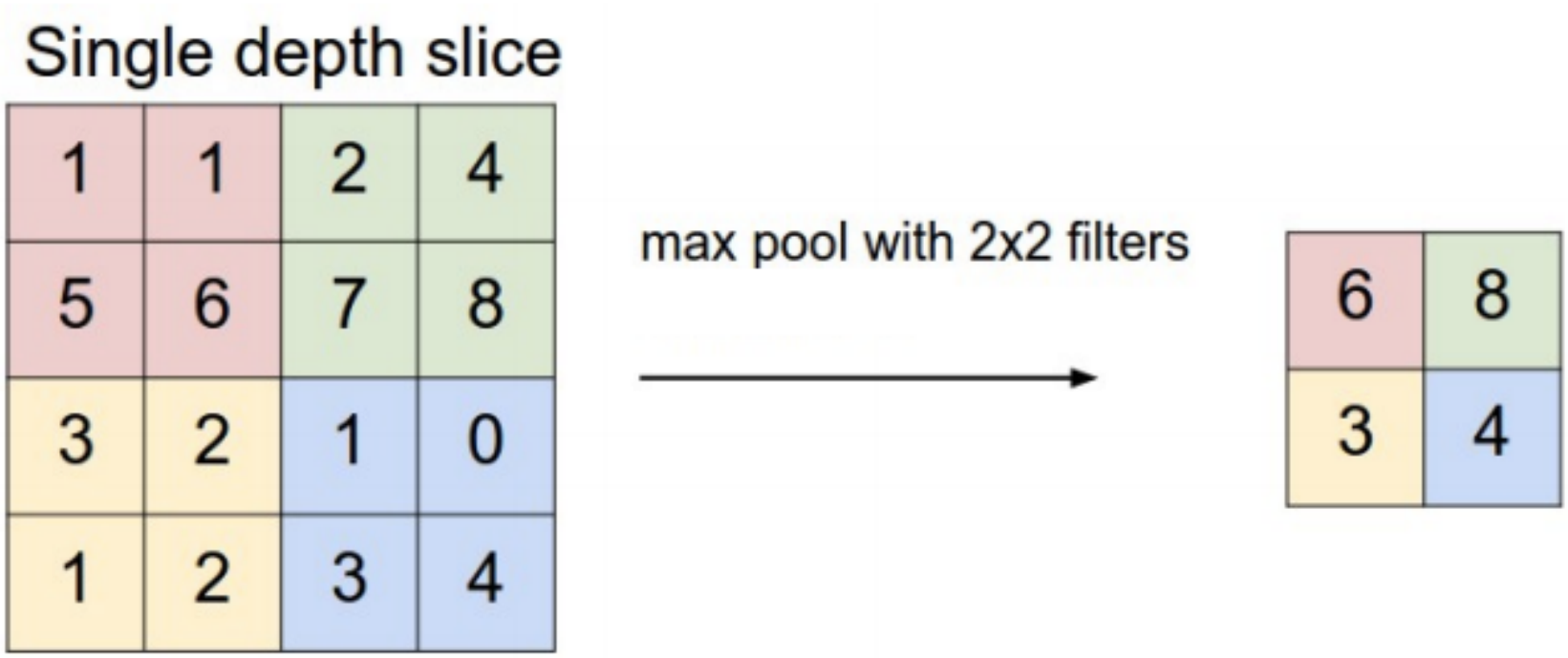

Figure 6: Max-pooling scheme.

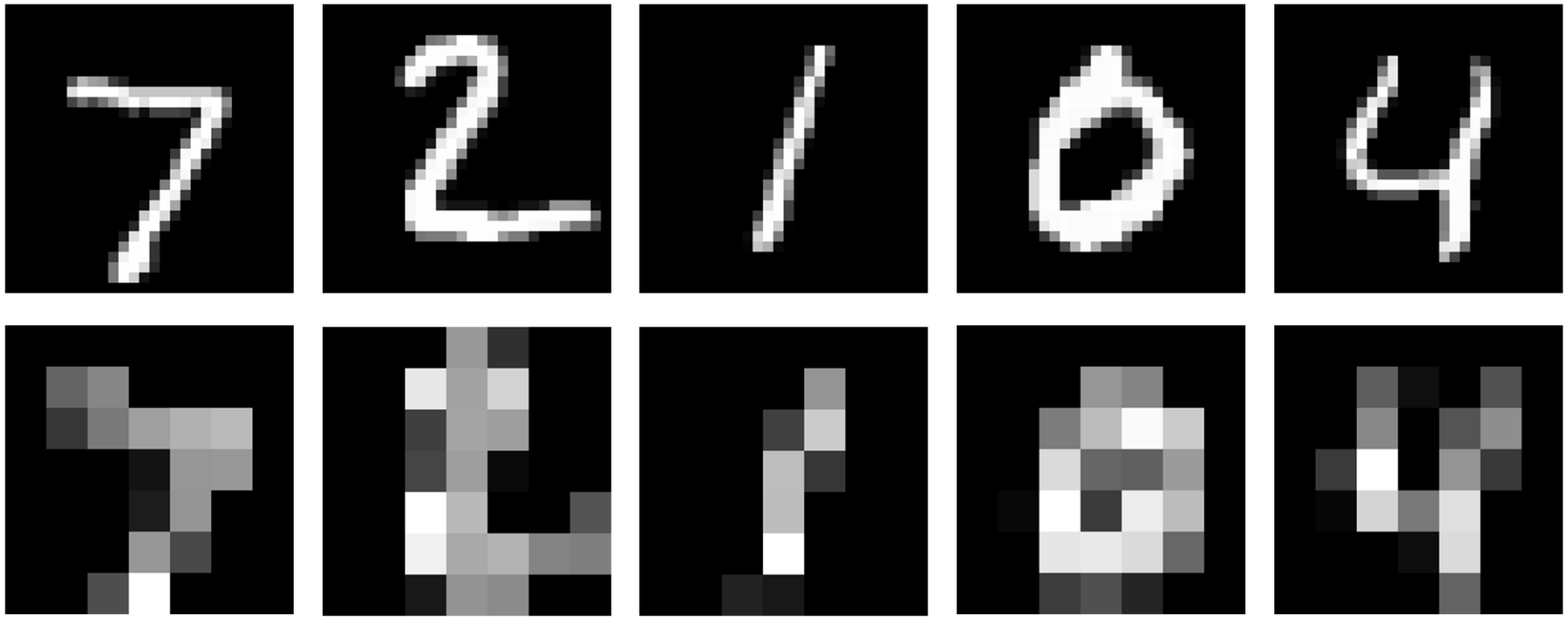

Figure 7: Max-pooling on 28x28 MNIST Data.

LeNet 5

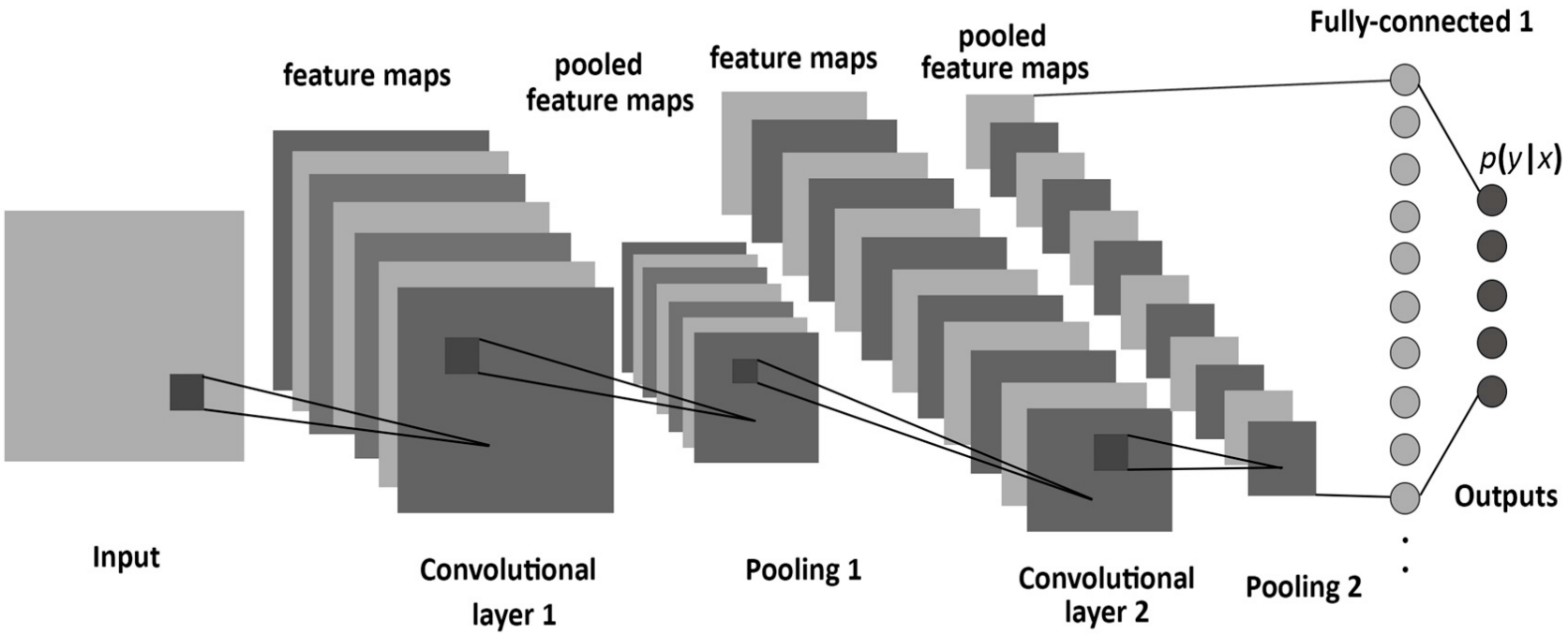

Figure 8: LeNet 5 architecture.

Learning weights

Cost function

$$J = -\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{j}[y_i \log \tilde{y}_i + (1 - y_i)\log(1 - \tilde{y}_i)] \tag{6}$$

Stochastic Gradient Descent (SGD)

$$\vec{w} \leftarrow \vec{w} - \epsilon_k \nabla_w J \tag{7}$$

Backpropagation

$$\frac{\partial J}{\partial w^l_{i,j}} = \sum_{x,y} a^{l-1}_{x+i,y+j}\delta^l_{x,y} \tag{8}$$

$$\delta^{l-1}_{x,y} = \sum_{i,j} \frac{\partial J}{\partial \xi^l_{x-i,y-j}} w_{i,j} f'(\xi^l_{x,y}) \tag{9}$$

Regularization

Dropout

$$r_j^l \sim \text{Bernoulli}(1, p) \tag{10}$$

$$\tilde{\vec{a}}^l = \vec{r}^l \circ \vec{a}^l \tag{11}$$

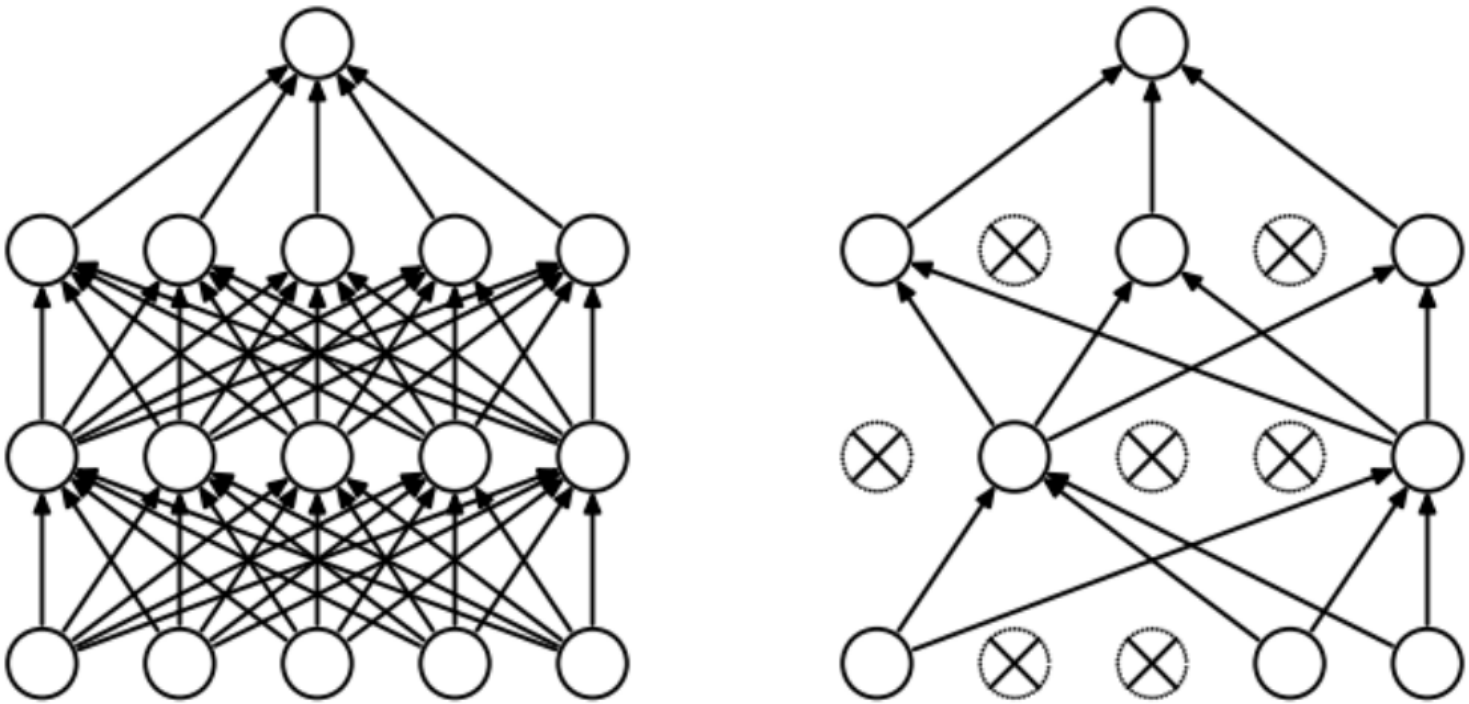

Figure 9: Visualisation of dropout regularization.

Regularization

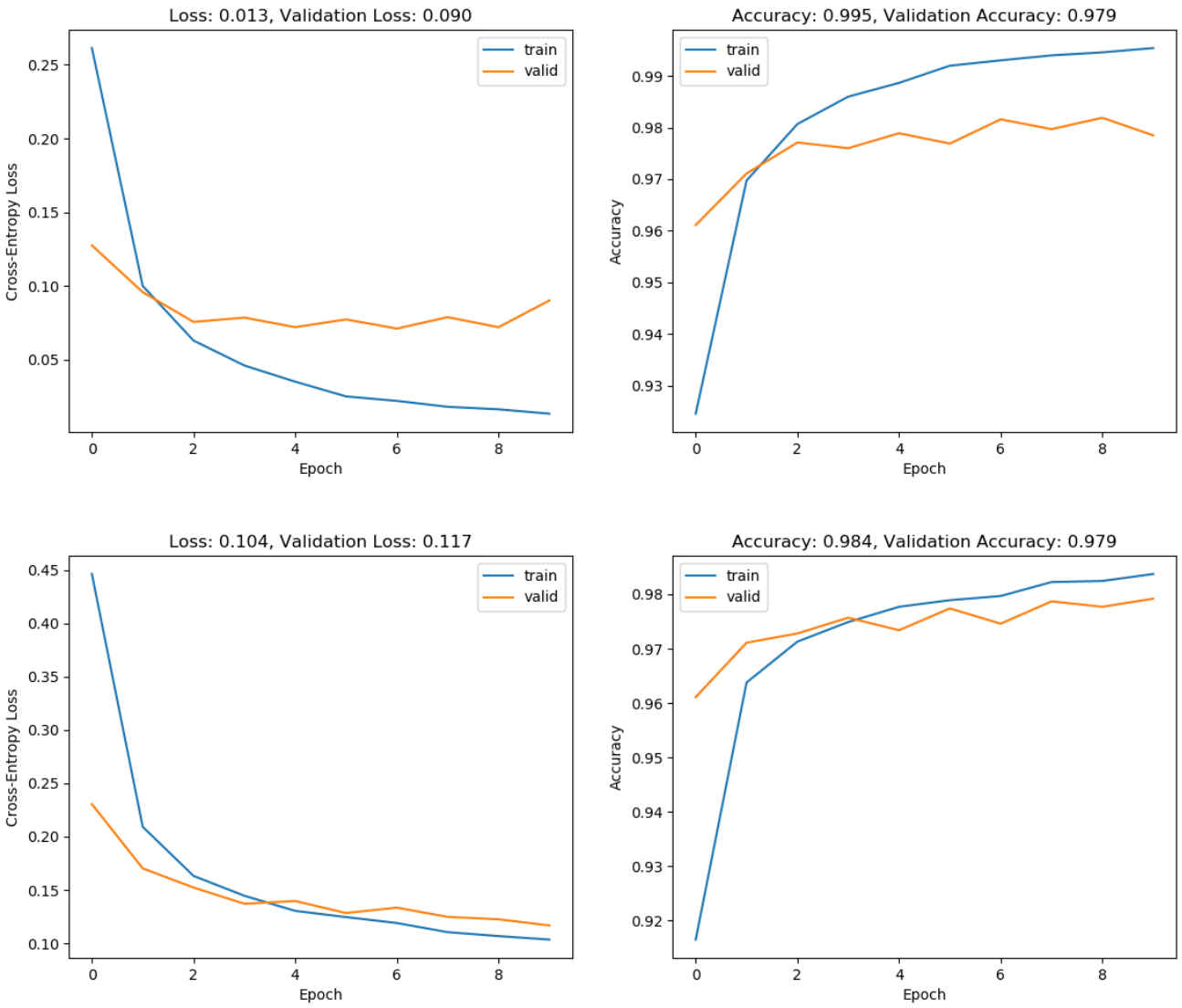

Figure 10: Effect of dropout on overtrained net.

Batch normalization

Let x_i represent values over a minibatch $B = \{x_1, \cdots, x_m\}$

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{12}$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{13}$$

$$\tilde{x}_i = \frac{x_i - \mu_B}{\sigma_B^2} \tag{14}$$

Normalized data allows networks to converge much faster

Spatial distortions

Generalization of dataset through data augmentation.

Figure 11: Elastic distortions.

Results on MNIST Dataset

Architecture	Error (%)
ANN (2-layer 784-800-10)	1.60
CNN (2-layer 32-128-10)	1.28
LeNet5 (6-layer 6-6-16-16-120-84-10)	0.80
CNN (5-layer 32-32-64-64-512-10)	0.55
CNN (5-layer 32-32-64-64-512-10, elastic distortions)	0.40
Comm. of 5 CNNs (6-layer 784-50-100-500-1000-10-10)	0.21

Visualisation of the results

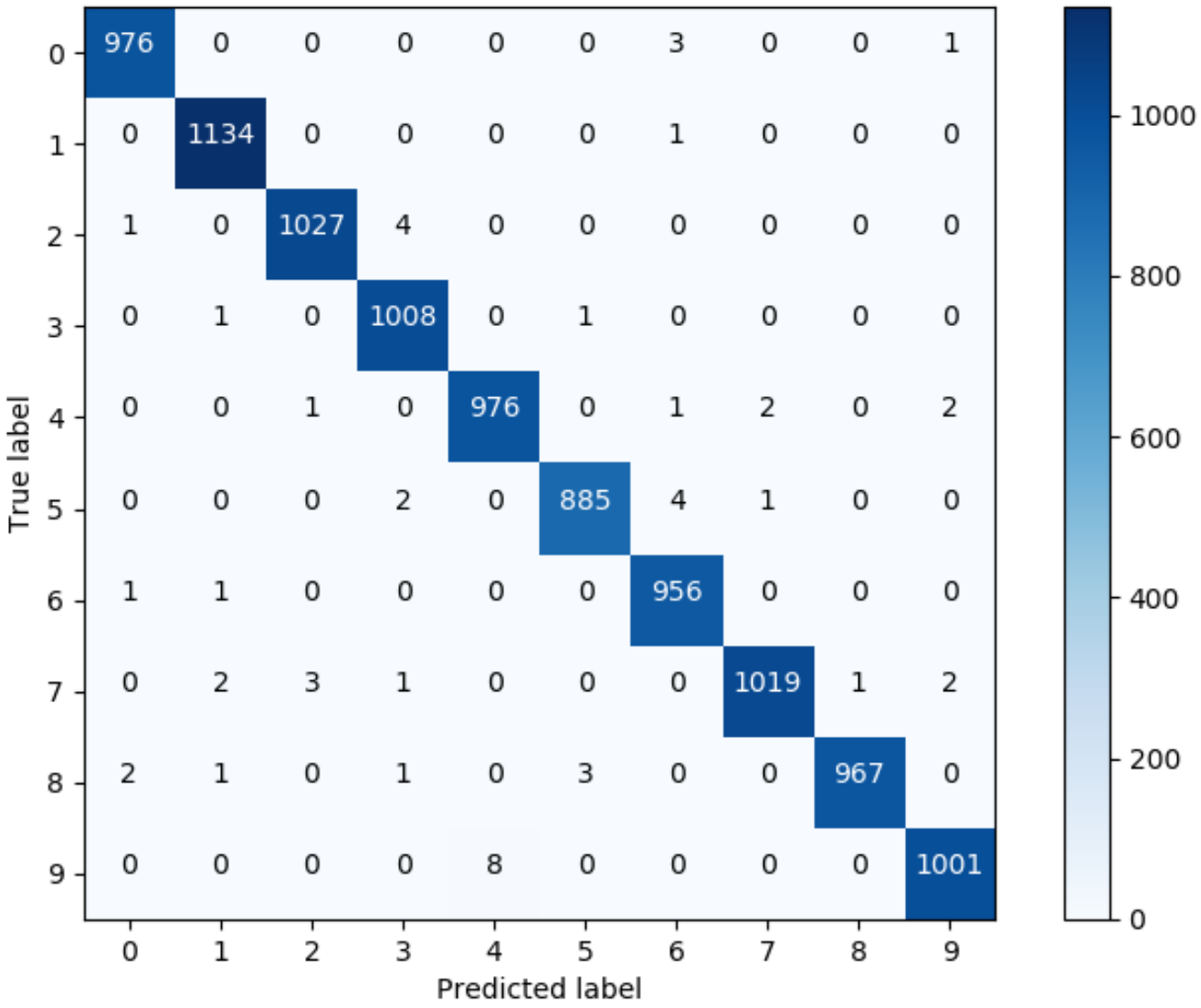

Figure 12: Confusion matrix for MNIST classification.

CNN in HEP

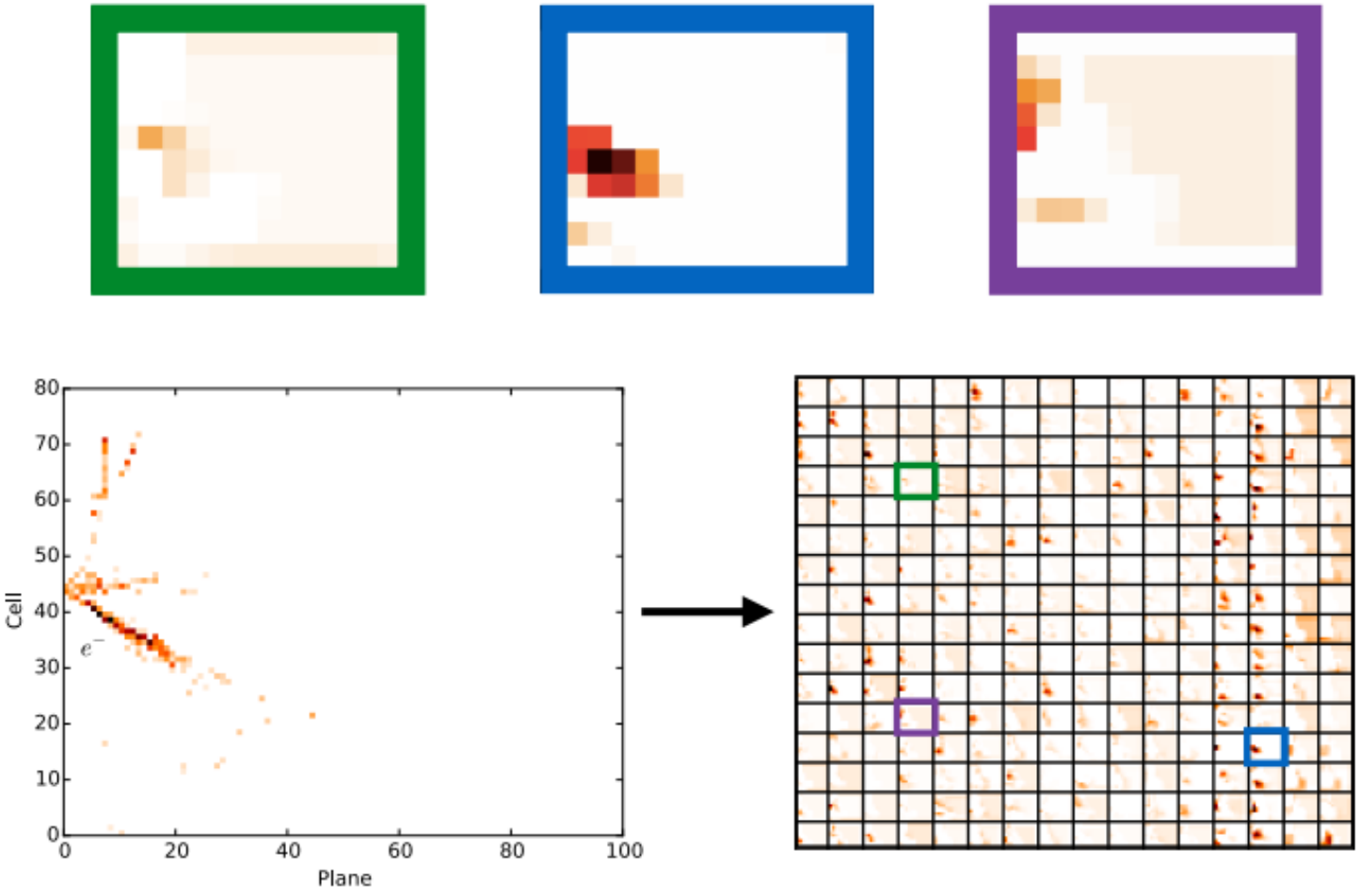

Figure 13: Detected features for different interactions.

Thank you for your attention

Literature

- Goodfellow-et-al-2016, Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press, 2016
- Christopher Bishop, Neural Networks for Pattern Recognition,Oxford University Press,1995
- Christopher Bishop, Pattern Recognition and Machine Learning,Oxford University Press,2006
- Karpathy-et-al,Convolutional Neural Networks for Visual Recognition, http://cs231n.github.io/convolutional-networks/, 2017