Neural Nets 12a,12b

Peter Savostin,
Arthur Martirosyan,
Jiang Lei

Content

Glossary

- **Axon** - аксон, отросток по которому идут нервные импульсы к нервным клеткам.
- **Dendrite** - дендрит, получает информацию через синапсы от аксонов.
- **Synaptic gaps** - синаптические разрывы между аксонами и дендритами.
- **Refactory period** - период рефрактерности, интервал, в течение которого возбудимая ткань не способна генерировать повторный сигнал.
- **Axonal bifurcation** - аксональная бифуркация, разделение аксона на 2 ветви одинакового диаметра, отходящие в стороны под одинаковыми углами.

Glossary

- **Threshold box** - пороговый блок, нейрон получает данные от других нейронов, определяет значение каждого входа и добавляет эти значения. Если общий вход выше пороговой величины, то выход блока равен единице, в противном случае – нулю.
- **Cumulative influence** - совокупное влияние нескольких входов нейрона
- **Synaptic weight** - синаптический вес, в биологии жидкость в синаптических разрывах

Glossary

- **Magnitude of difference** - порядок величины
- **Performance function** - функция оценки качества работы нейронной сети
- **Computationally intractable** - вычислимо сложно
- **Gradient ascent** - градиентное восхождение
- **Chain Rule** - правило дифференцирования сложной функции

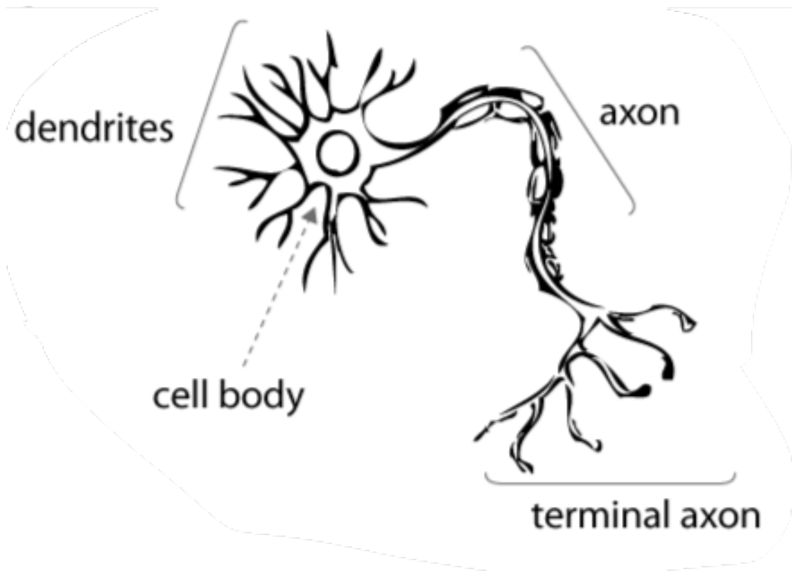
Glossary

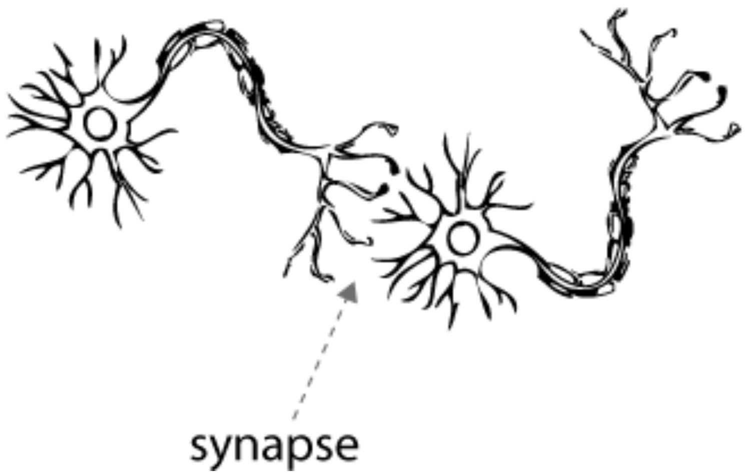
- **Dot product** - скалярное произведение
- **Deep neural nets** - глубокие нейронные сети, сеть с несколькими скрытыми слоями нейронов
- **Convolutional neural network** - сверточная нейронная сеть
- **Pooling** - объединение, одна из стадий работы сверточной нейронной сети
- **Kernel** - ядро, функция применяемая к пикселям некоторого изображения
- **Vivinity** - окрестности

Glossary

- **Grid** - сетка
- **Back propogation** - метод обратного распространения ошибки
- **Generalization** - обощение
- **Boltzmann machines** - вид рекуррентных нейронных сетей
- **Softmax** - функция, «сжимает» вектор действительных величин до вектора с элементами от 0 до 1
- **Saddle points** - седловые точки

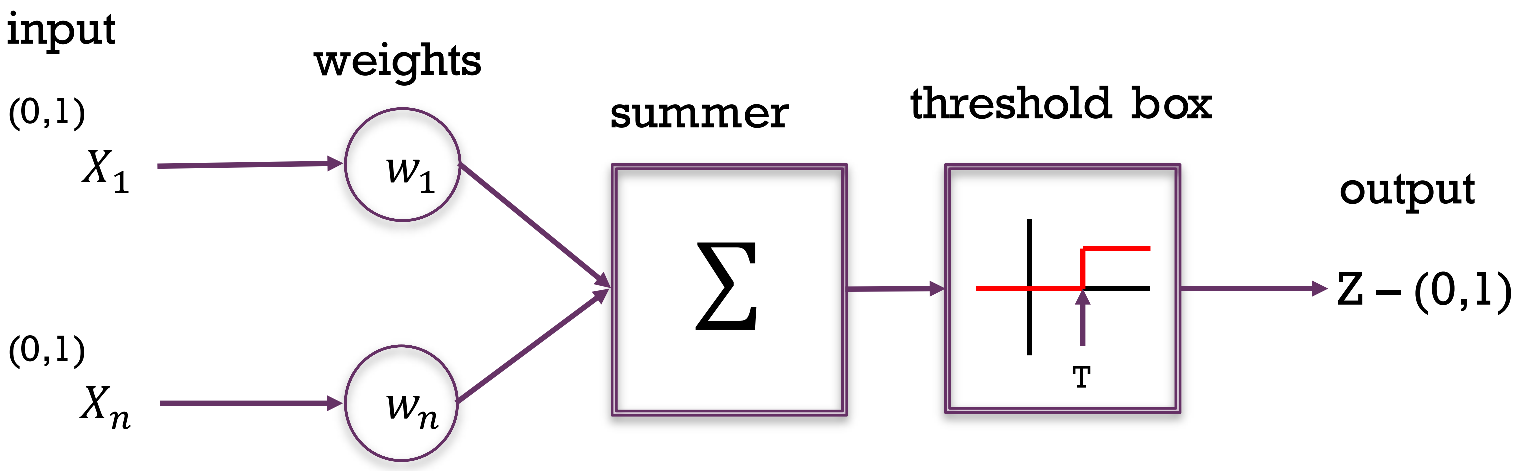
Biological model





1. All or non
2. Cumulative influence
3. Synaptic weight
4. Refractor period
5. Axonal bifurcation
6. Time patterns

Model inspired by our brain



What we can model:

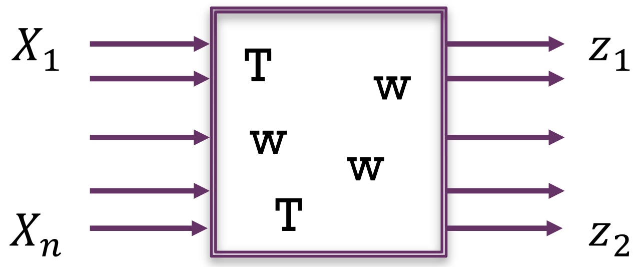
1. All or non – threshold box
2. Cumulative influence - summer
3. Synaptic weight - weights

What we don't know how to model:

4. Refractor period
5. Axonal bifurcation
6. Time patterns

Neural net as function approximator

Our skull is like a "box full of neurons", in fact it is better to say "box full of weights and thresholds"



Then output is a function of input vector, weight vector and threshold vector

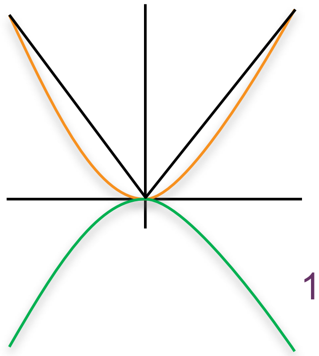
$$\bar{z} = f(\bar{x}, \bar{w}, \bar{T})$$

When we will train neural net, all we are going be able to do is adjust $\bar{w}, \bar{T}$, that's why neural net is some kind of function approximator

We can figure out how good our neural net is by using performance function P.
It compares the desired value and the actual value

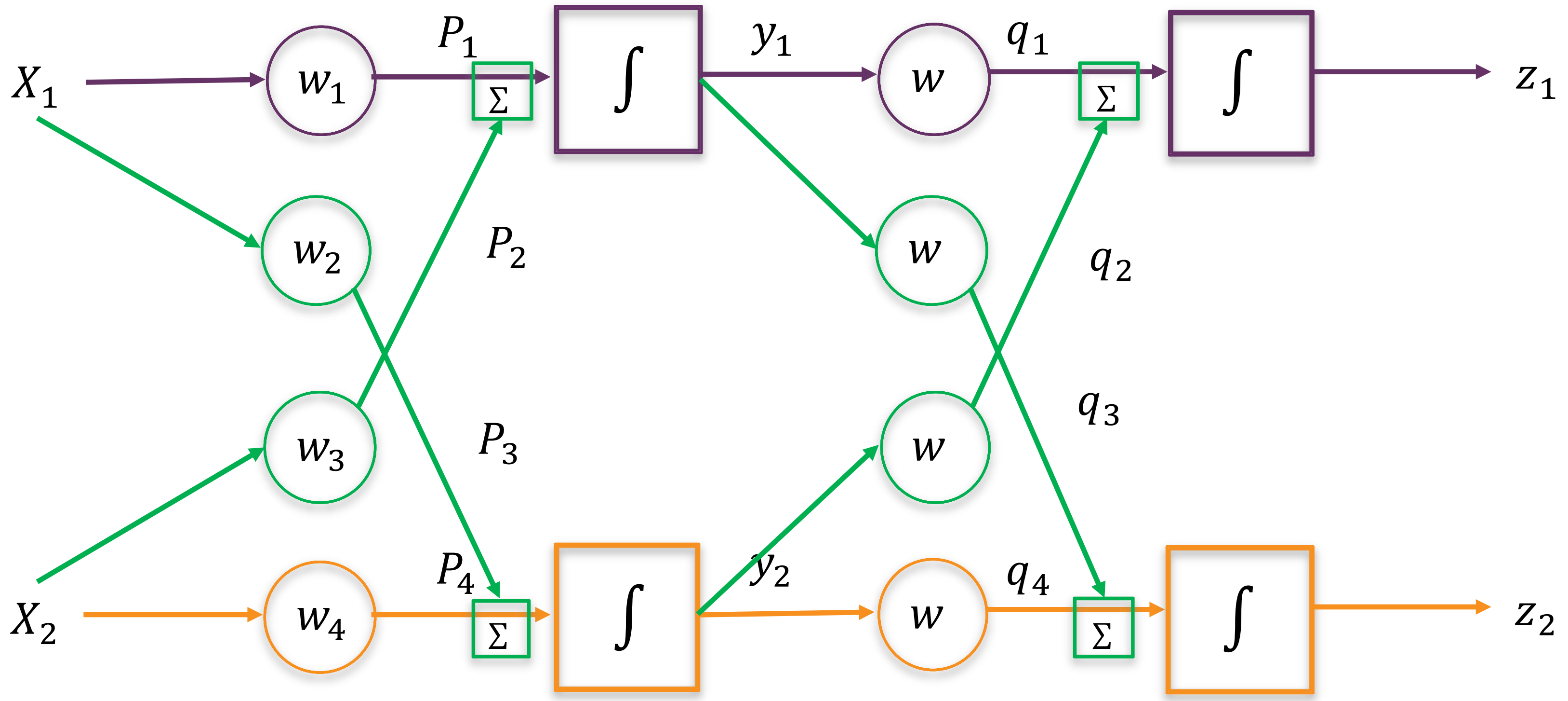
$$P = -\|\bar{d} - \bar{z}\|^2$$

$$\bar{d} = g(\bar{x})$$



Complexity of cross connected neural net

We can add multiple layers and make cross connections between nodes.
Will it affect the complexity and this network can be growing exponential?



Savostin, Martirosyan

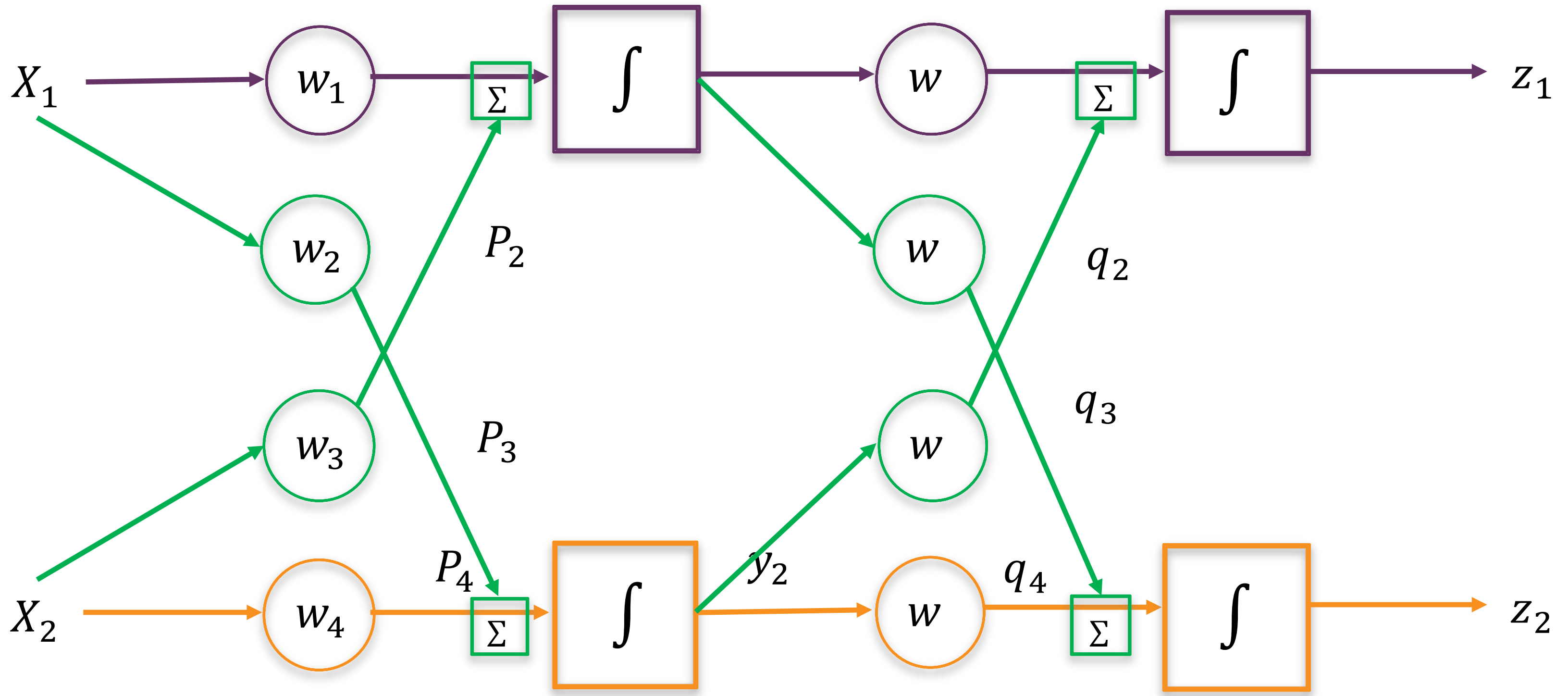
Complexity of cross connected neural net

$$\frac{\partial P}{\partial w_1} = \frac{\partial P}{\partial z_1}\frac{\partial z_1}{\partial q_1}\frac{\partial q_1}{\partial y_1}\frac{\partial y_1}{\partial p_1}\frac{\partial p_1}{\partial w_1} + \frac{\partial P}{\partial z_2}\frac{\partial z_2}{\partial q_2}\frac{\partial q_2}{\partial y_1}\frac{\partial y_1}{\partial p_1}\frac{\partial p_1}{\partial w_1}$$

$$\frac{\partial P}{\partial w_3} = \frac{\partial P}{\partial z_1}\frac{\partial z_1}{\partial q_1}\frac{\partial q_1}{\partial y_1}\frac{\partial y_1}{\partial p_3}\frac{\partial p_3}{\partial w_3} + \frac{\partial P}{\partial z_2}\frac{\partial z_2}{\partial q_2}\frac{\partial q_2}{\partial y_1}\frac{\partial y_1}{\partial p_3}\frac{\partial p_3}{\partial w_1}$$

Width - w^2

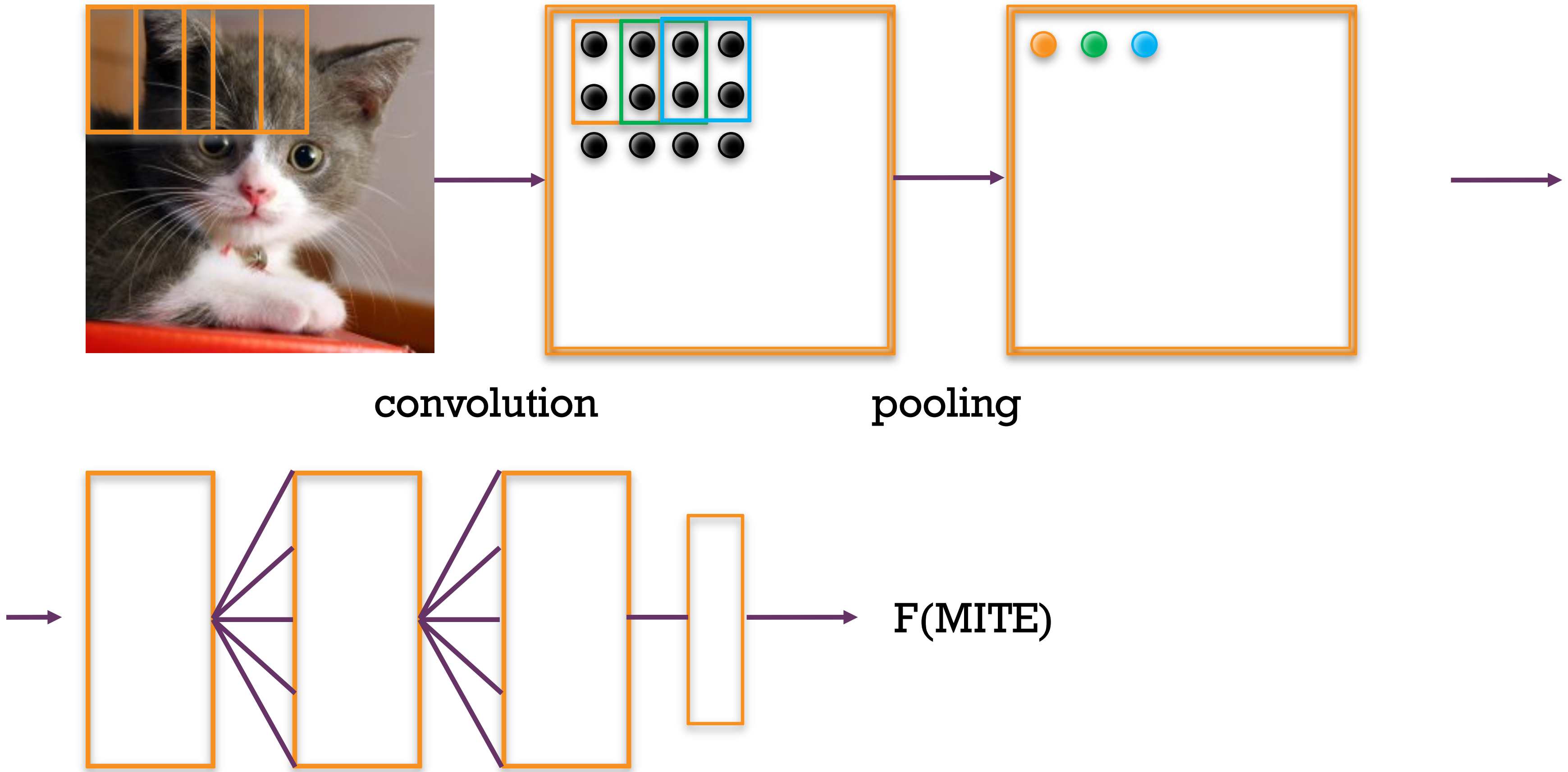
Linear in depth



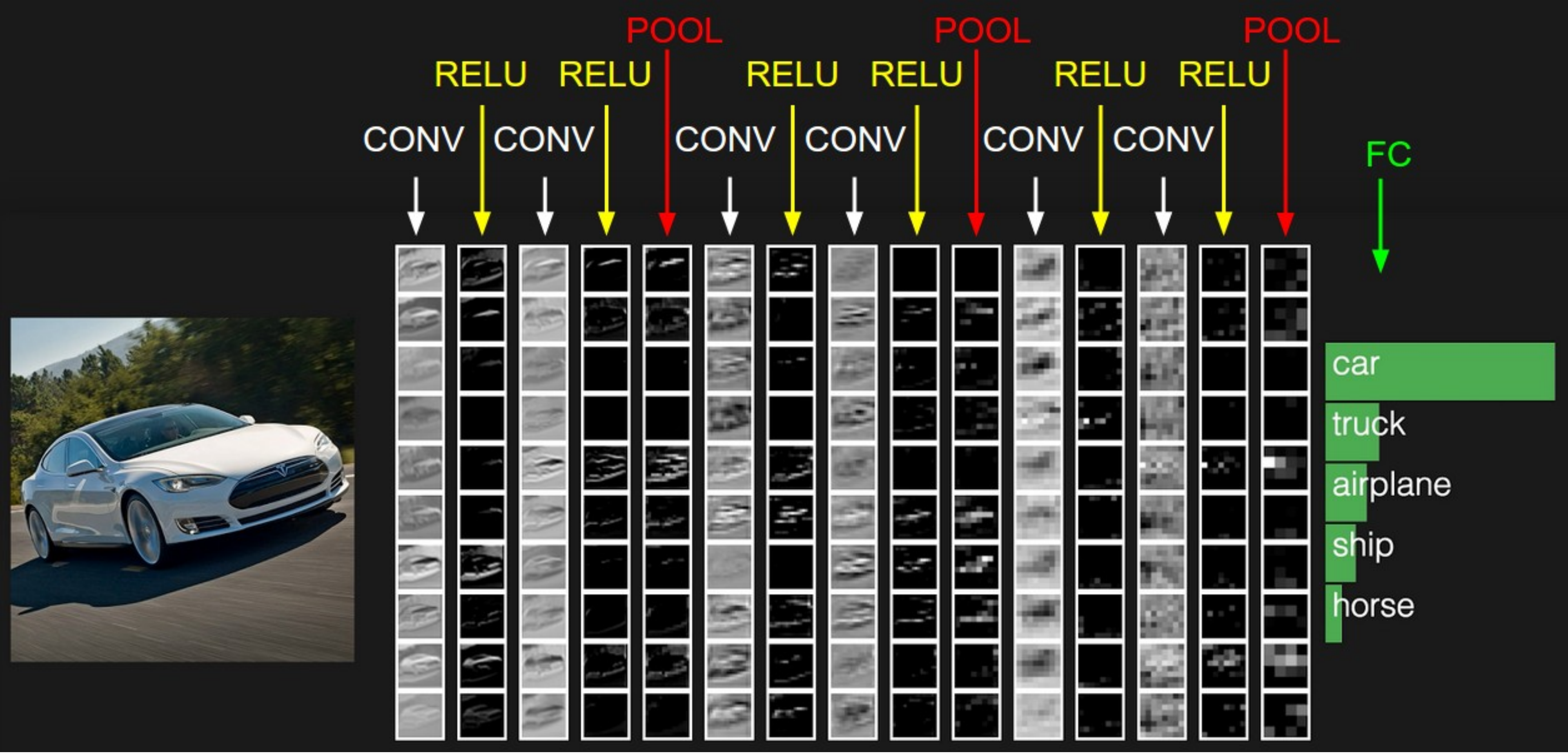
Savostin, Martirosyan

Demonstration

Convolutional Neural Networks

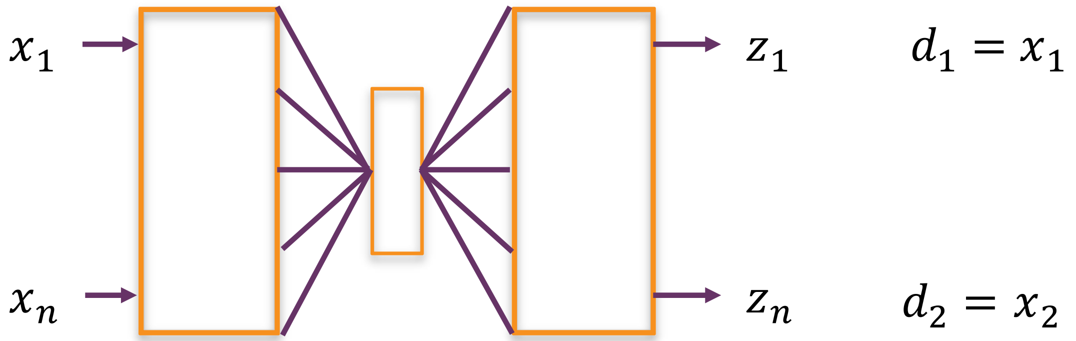


Savostin, Martirosyan



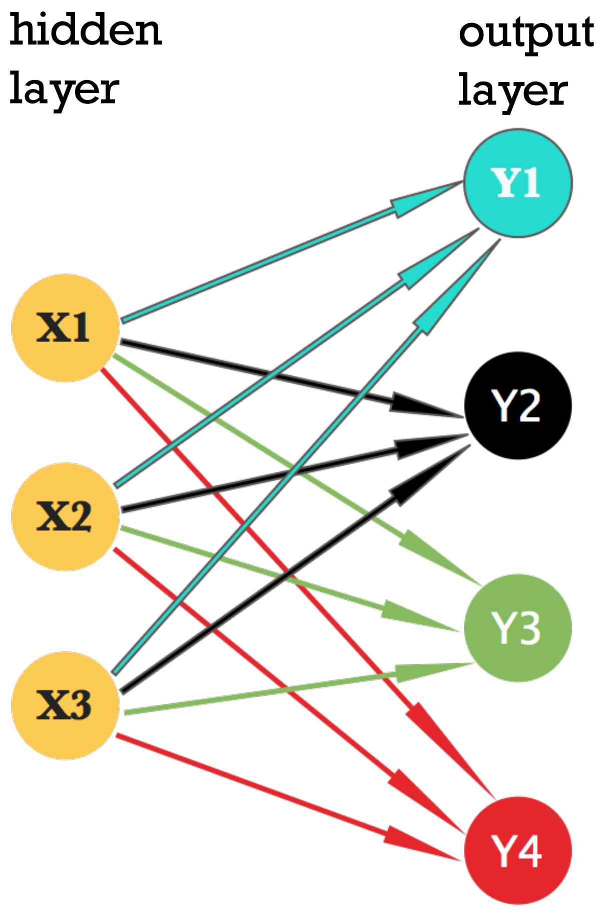
INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
RELU layer will apply an elementwise activation function, such as the max(0,x)max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).

Autocoding



Network has a tight bottleneck of a few neurons in the middle, forcing it to create effective representations that compress the input into a low-dimensional code that can be used by the decoder to reproduce the original input.

Final Layer



In the final layer, we want to associate a probability with each output Y_i. Through sigmoid function, we can get a $F_i(X)$, how can we get the probability? Just use the formula:

$$Y_i = P_i(X) = \frac{F_i(X)}{\sum_{j=1} F_j(X)}$$

So that we can get a probability for each output, it's a extension of sigmoid function.

Jiang Lei

Softmax

Softmax function is a formula like this:

$$P(i) = \frac{\exp(\theta_i^T x)}{\sum_{k=1}^{K} \exp(\theta_k^T x)}$$

Through softmax, we can make $P(i)$ belongs to range $[0,1]$. In the classification problem, θ is the parameters we want to get to make $P(i)$ as big as possible. We have many ways to make $P(i)$ belongs to $[0,1]$, why here we use $\exp()$? Refer to logistic function, we can use this function to make the positive result tend to 1 and make the negative result tend to 0.

Jiang Lei

Question

What the relation between softmax and sigmoid function?

In fact, softmax is a generalization of the logistic function. As we know, we can use logistic function to do 2-dimensional classification. In the same way, we can use softmax to do k-dimensional classification. So, we can say:

Softmax function is a generic sigmoid function.

Sigmoid function is a specific softmax function.

Jiang Lei

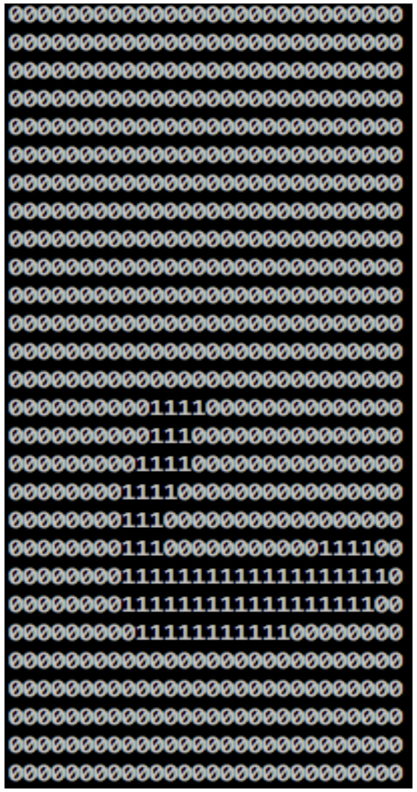
Dropout

In the class, this part of explanation is not very clear, we can use an example to review dropout.

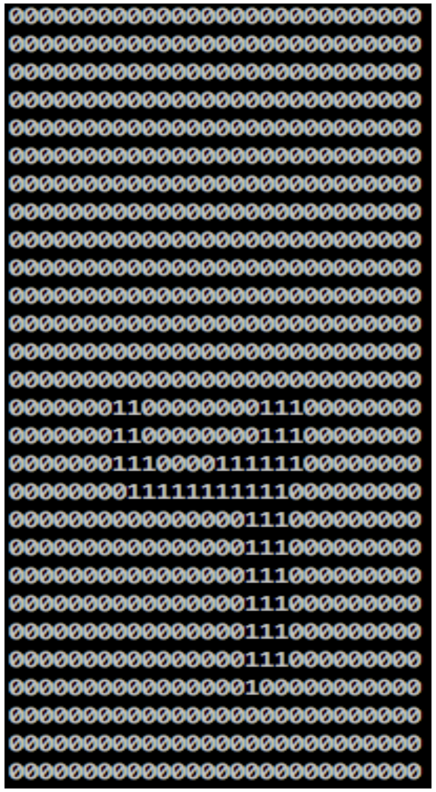
Some of us maybe know about MNIST database of handwritten digits. We can train a neural network to recognize them. When we test the model, we don't use the whole image(digit), we cover half of the digit, like the following:

Jiang Lei

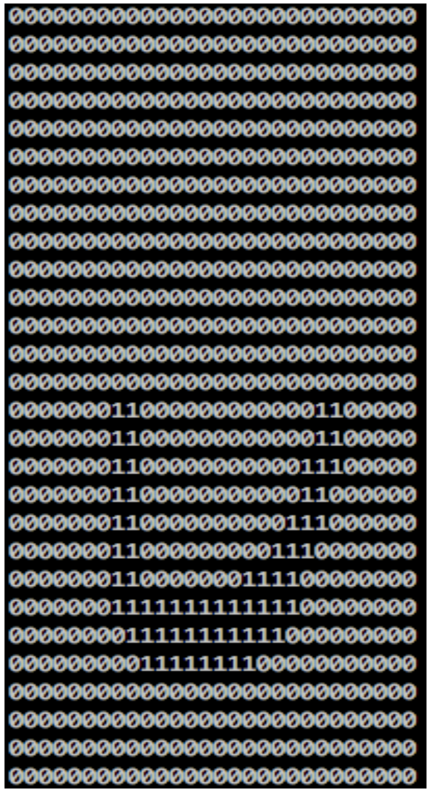
Half-Digit



```
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000111100000000000000
0000000000111000000000000000
0000000001111000000000000000
0000000011110000000000000000
0000000011100000000000000000
0000000011100000000000111100
0000000011111111111111111110
0000000011111111111111111100
0000000001111111111110000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
```



```
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000110000000011100000000
0000000110000000011100000000
0000000111000011111100000000
0000000011111111111100000000
0000000000000000111000000000
0000000000000000111000000000
0000000000000000111000000000
0000000000000000111000000000
0000000000000000111000000000
0000000000000000111000000000
0000000000000000100000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
```



```
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000110000000000001100000
0000000110000000000001100000
0000000110000000000011100000
0000000110000000000011000000
0000000110000000000111000000
0000000110000000001110000000
0000000110000000111100000000
0000000111111111111100000000
0000000011111111111000000000
0000000001111111100000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
```

Jiang Lei

Inference

We can find that, even if for us humans, it is difficult to identify them. If we cover part of the pictures, how can we identify them? For us humans, as long as it is not to cover the most critical information, human beings can identify some of the whole object through local information. This shows that humans have the ability to use partial information for inference. How can we make the neural network have such an ability? It is Dropout.

Dropout

At this time, we may use dropout idea to solve the problem. In the training process, I only use part of the neural units, not all units. That means I will drop out some information in every training step. Because every time I only use part of the training information, then I naturally have the ability of using part of the information to predict, so that our model become more generalization.

Jiang Lei

Question

A great question from reddit:

Why do I never see dropout applied in convolutional layers?

The additional gain in performance obtained by adding dropout in the convolutional layers is worth noting. One may have presumed that since the convolutional layers don't have a lot of parameters, overfitting is not a problem and therefore dropout would not have much effect. However, dropout in the lower layers still helps because it provides noisy inputs for the higher fully connected layers which prevents them from overfitting.

Jiang Lei