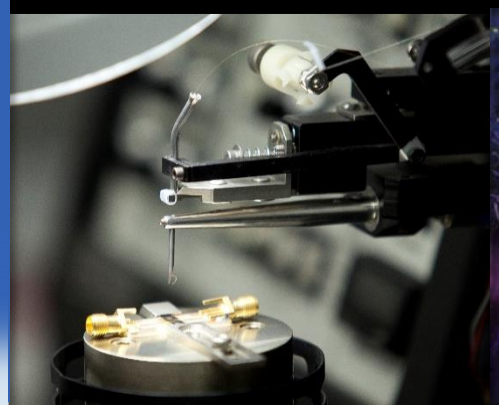
Centro Brasileiro de Pesquisas Físicas

Redes Neurais profundas e aplicações
Deep Learning

Clécio Roque De Bom – debom@cbpf.br

clearnightsrthebest.com

CBPF






Regularization

What´s for? Also prevents overfitting. Adds a Contrain.

Consider a simple Loss like residual sum of squares:

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

One can add a penalty:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

This is known as L2 regularization

https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

Regularization

What´s for? Also prevents overfitting. Adds a Contrain.

Consider a simple Loss like residual sum of squares:

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

One can add a penalty:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

This is known as L2 regularization

Warning: The coefficients are scale equivariants with the inputs, this proprierty is lost when regularizing, so one should standardize the inputs.

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2}}.$$

https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

Regularization

What´s for? Also prevents overfitting. Adds a Contrain.

Consider a simple Loss like residual sum of squares:

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

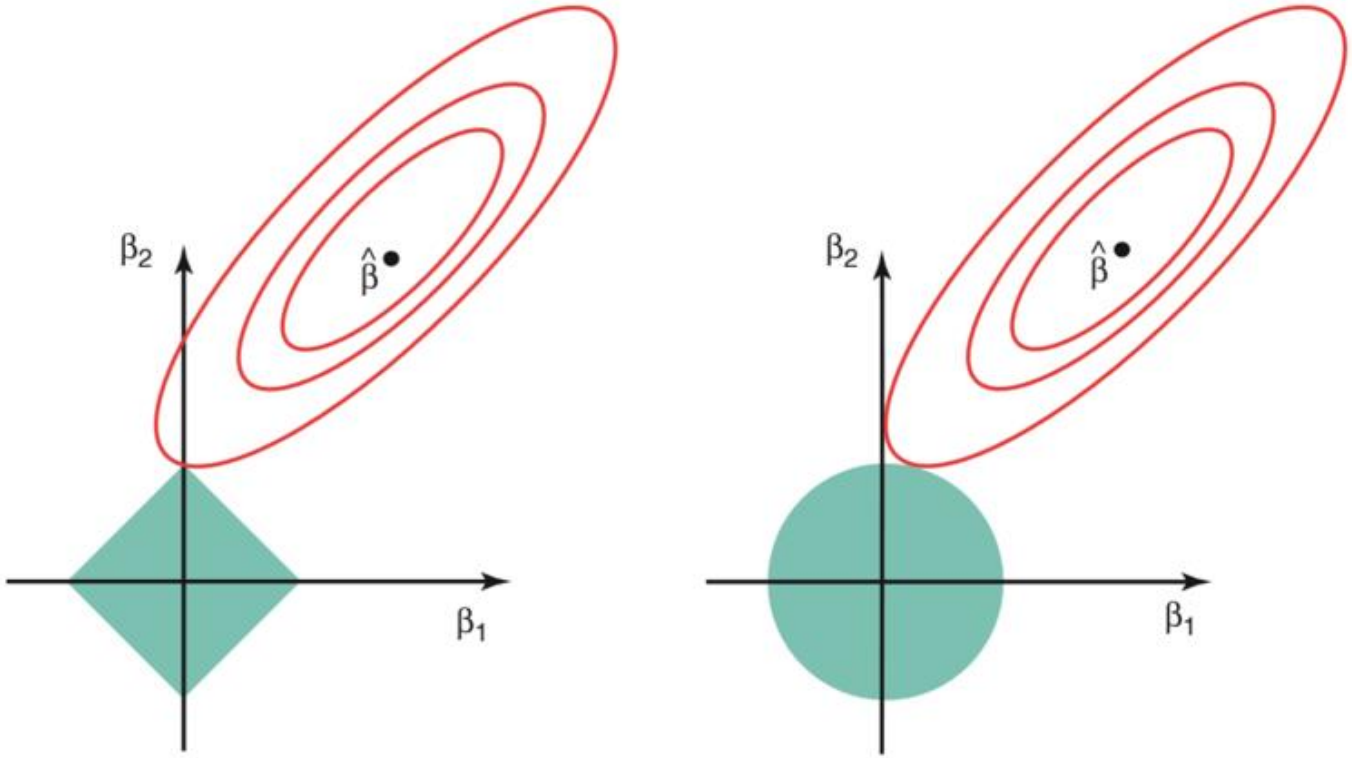
One can add a penalty:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = \mathrm{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.$$

This is known as L1 regularization

https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

Regularization



L1

Sparsity

L2

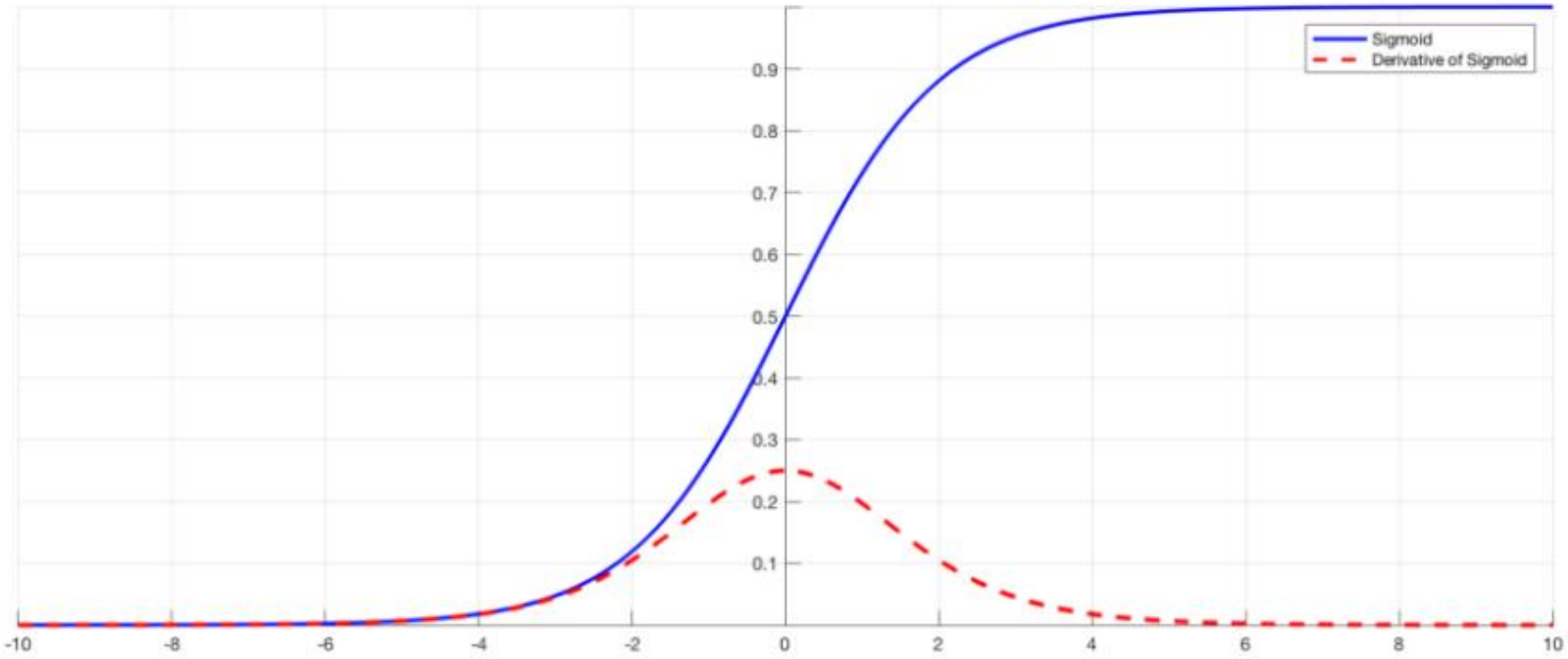
It potentially reduces the deviations from different training sets.

Credit : An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

The above image shows the constrain

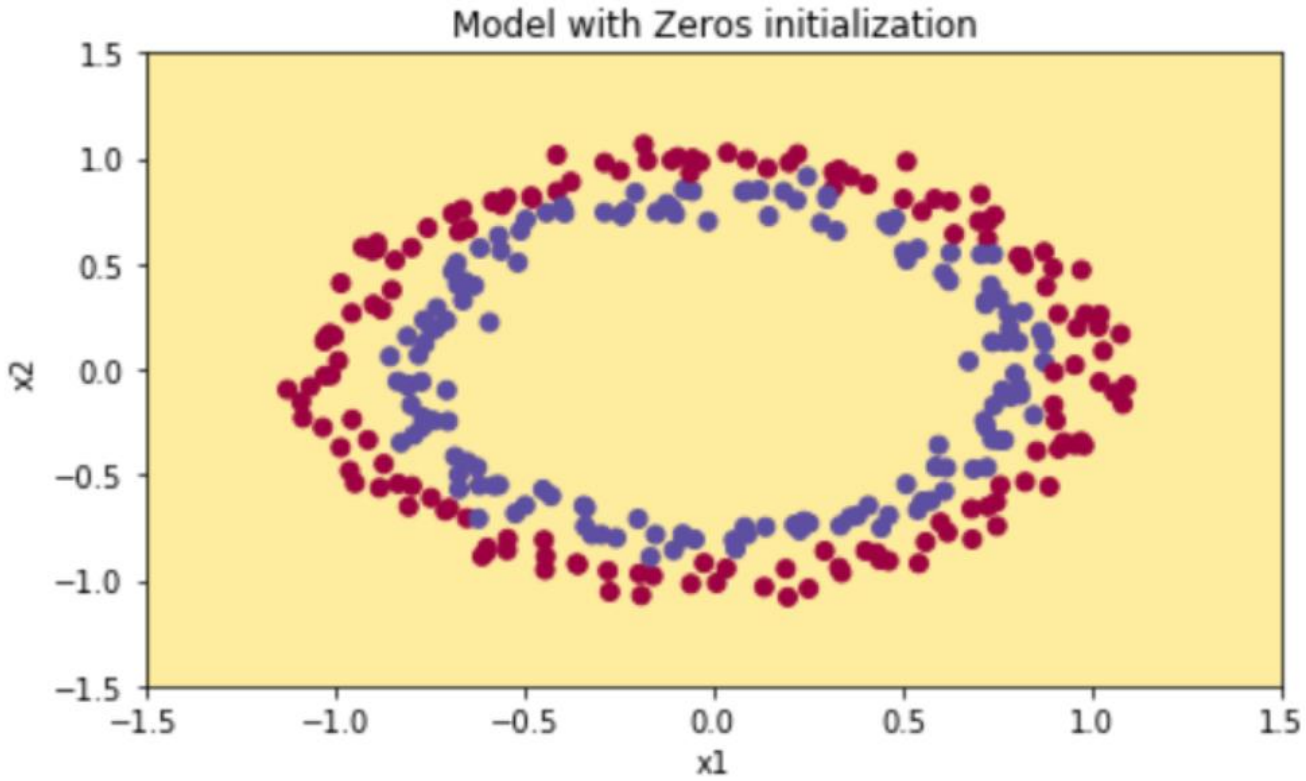
Initialization

0 initialization, all derivatives are the same ..., no matter the input is like.



Initialization

Example, 3 layers with ReLu activation function and sigmoid for the output layer



https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78

Random Initialization

Random

```
model.add(Dense(64, kernel_initializer='random_uniform',
bias_initializer='zeros'))
```

a) If weights are initialized with very high values the term np.dot(W,X)+b becomes significantly higher and if an activation function like sigmoid() is applied, the function maps its value near to 1 where the slope of gradient changes slowly and learning takes a lot of time.
b) If weights are initialized with low values it gets mapped to 0, where the case is the same as above.

https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78

Random Initialization

Random

```
model.add(Dense(64, kernel_initializer='random_uniform',
bias_initializer='zeros'))
```

a) If weights are initialized with very high values the term np.dot(W,X)+b becomes significantly higher and if an activation function like sigmoid() is applied, the function maps its value near to 1 where the slope of gradient changes slowly and learning takes a lot of time.
b) If weights are initialized with low values it gets mapped to 0, where the case is the same as above.

ReLU and LeakyReLu are your friends

Random Initialization – going deeper

Xavier Initialization

The normalization factor may therefore be important when initializing deep networks because of the multiplicative effect through layers, and we suggest the following initialization procedure to approximately satisfy our objectives of maintaining activation variances and back-propagated gradients variance as one moves up or down the network. We call it the **normalized initialization**:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010.

Random Initialization – going deeper

Xavier Initialization

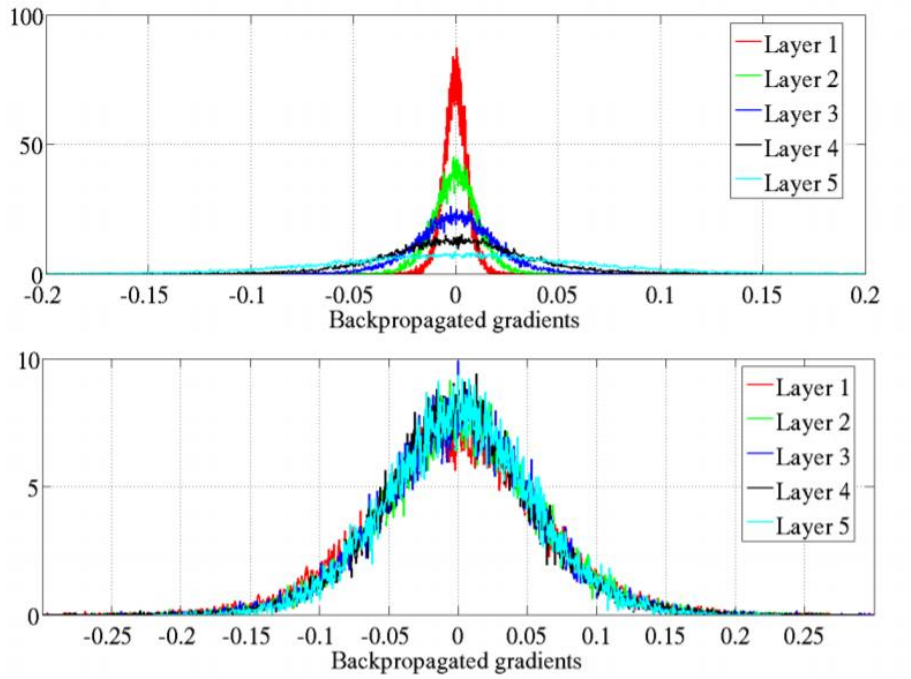


Figure 7: *Back-propagated gradients normalized histograms with hyperbolic tangent activation, with standard (top) vs normalized (bottom) initialization. Top: 0-peak decreases for higher layers.*

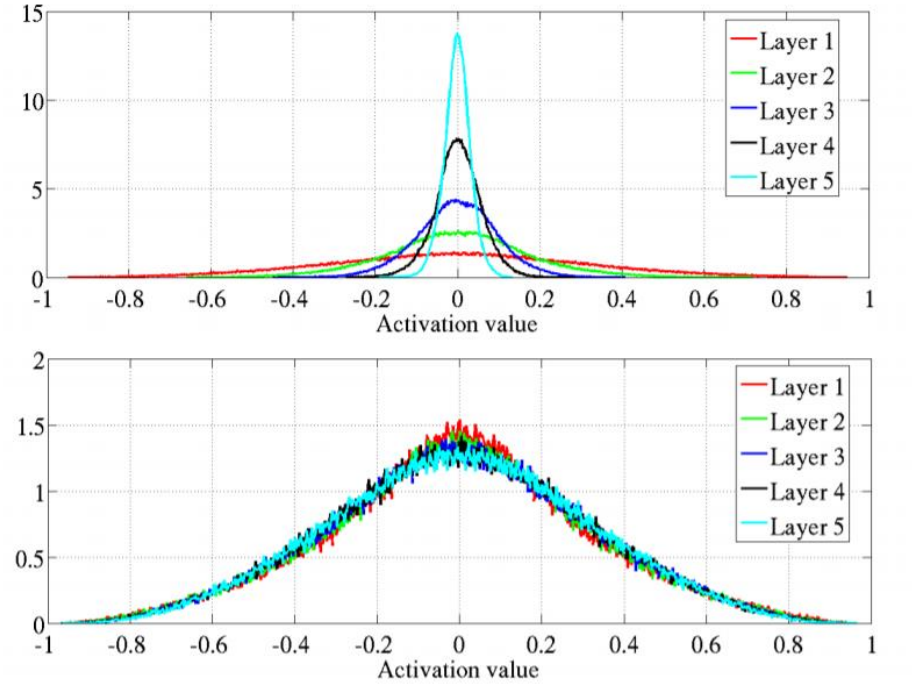


Figure 6: *Activation values normalized histograms with hyperbolic tangent activation, with standard (top) vs normalized initialization (bottom). Top: 0-peak increases for higher layers.*

Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010.

Random Initialization – going deeper

Xavier Initialization

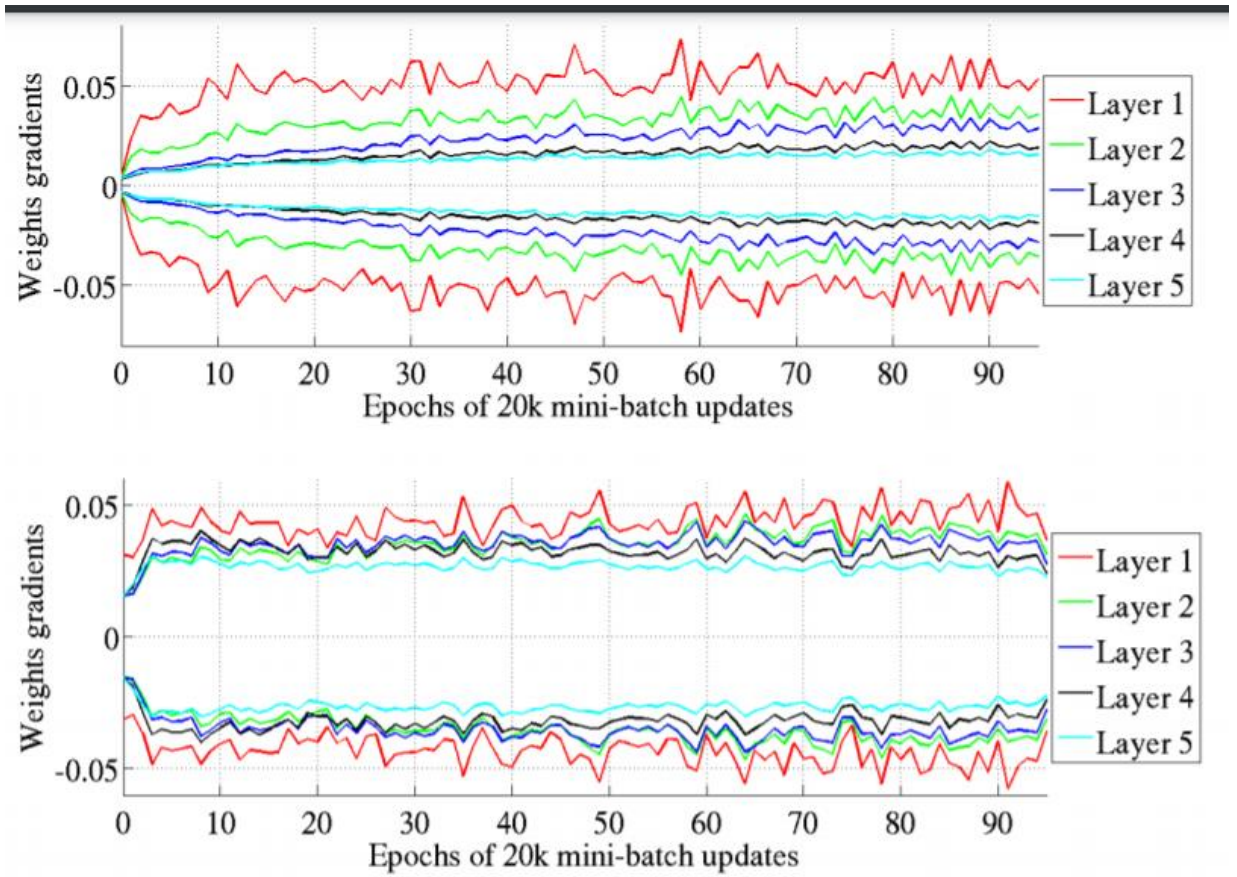
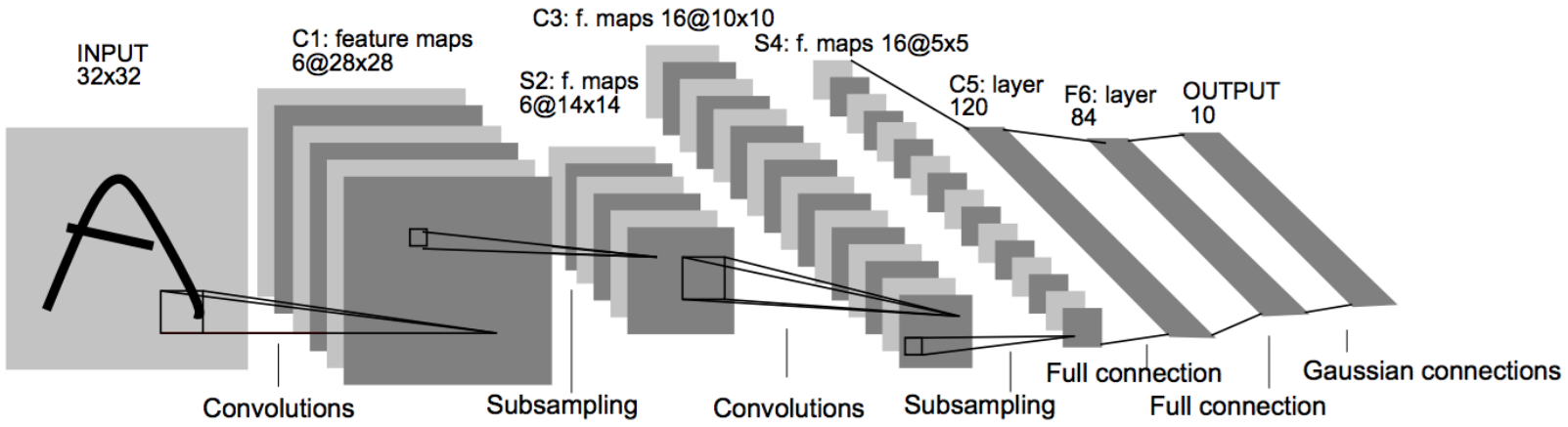


Figure 9: *Standard deviation intervals of the weights gradients with hyperbolic tangents with standard initialization (top) and normalized (bottom) during training. We see that the normalization allows to keep the same variance of the weights gradient across layers, during training (top: smaller variance for higher layers).*

Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010.

A little bit of historical Nets ...

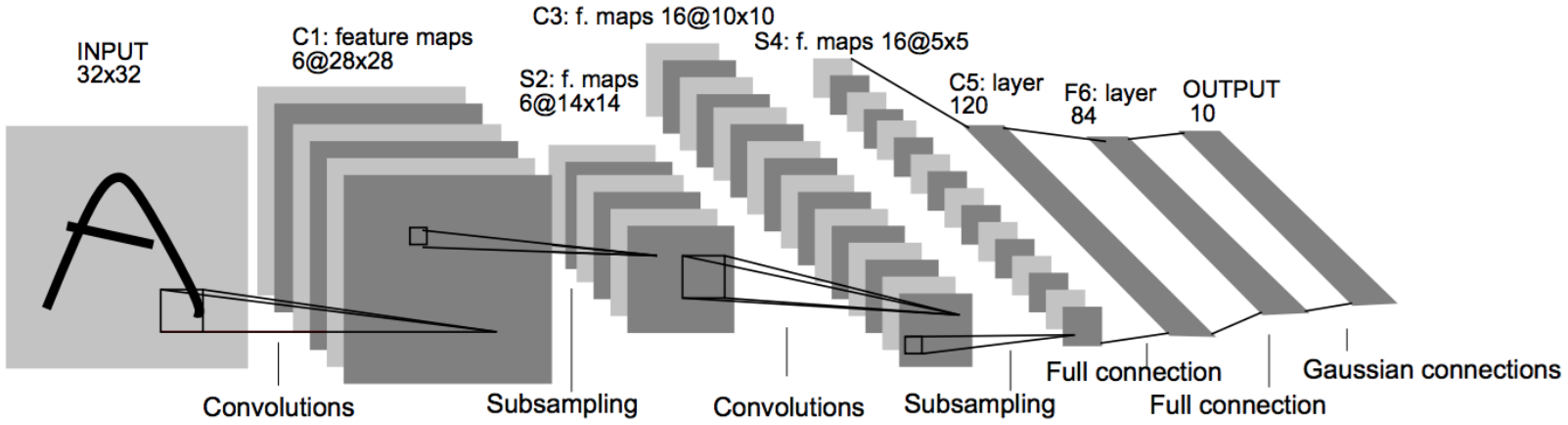
LeNet-5
Yann Lecun's LeNet-5 model was developed in 1998 to identify handwritten digits for zip code recognition in the postal service. This pioneering model largely introduced the convolutional neural network as we know it today.



LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

A little bit of historical Nets ...

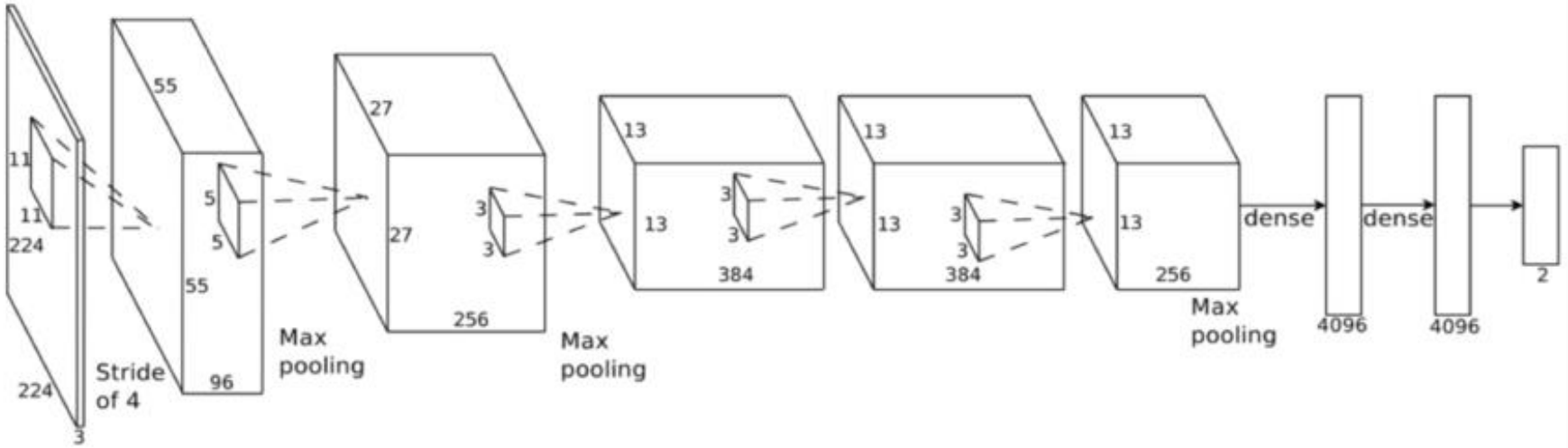
The subsampling layers use a form of average pooling.
Parameters: 60,000



LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

A little bit of historical Nets ...

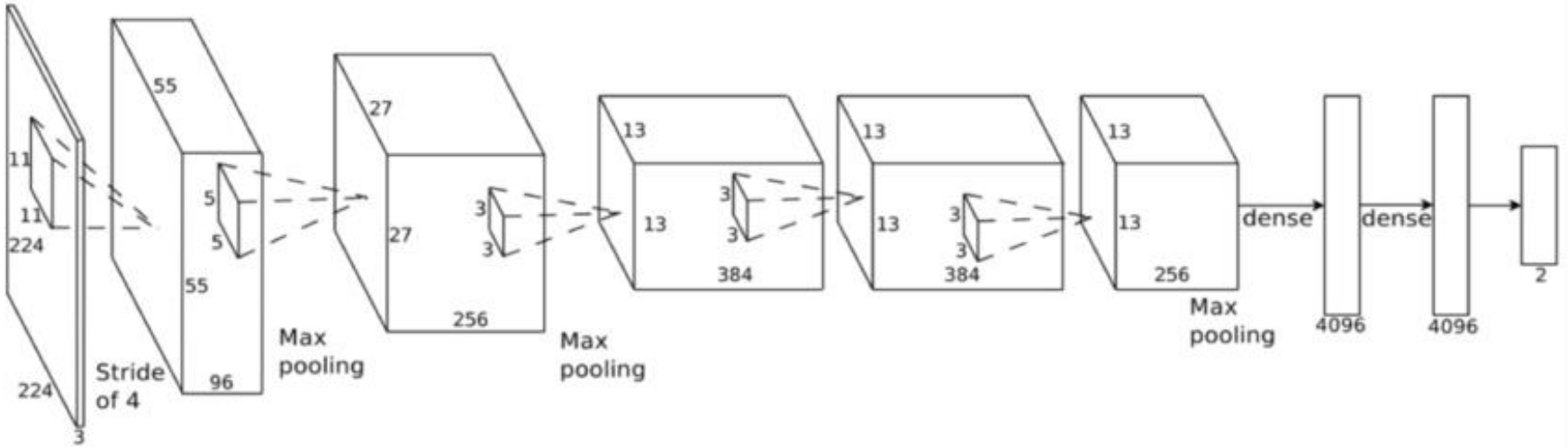
AlexNet was developed by Alex Krizhevsky et al. in 2012 to compete in the ImageNet competition. The general architecture is quite similar to LeNet-5, although this model is considerably larger. The success of this model (which took first place in the 2012 ImageNet competition) convinced a lot of the computer vision community to take a serious look at deep learning for computer vision tasks.



A little bit of historical Nets ...

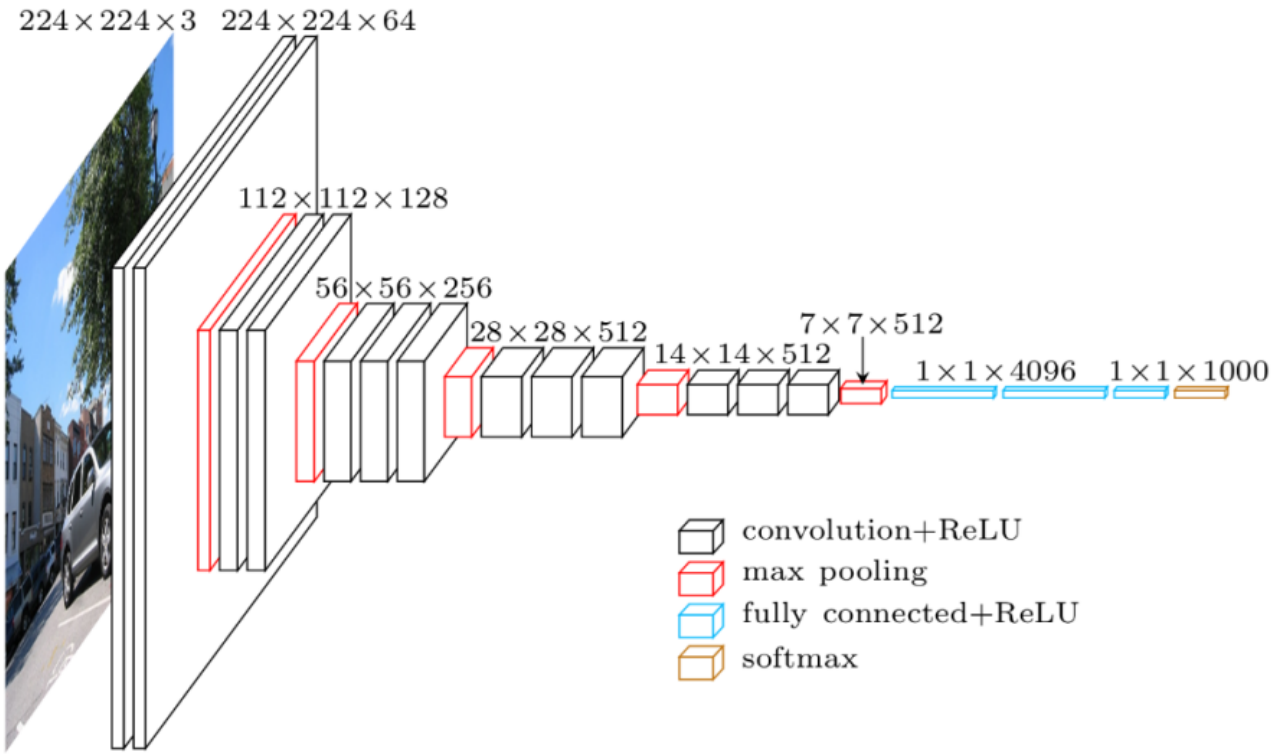
AlexNet → 60 Million parameters!!!!

650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three globally-connected layers with a final 1000-way softmax. It was trained on two NVIDIA GPUs for about a week.



A little bit of historical Nets ...

VGG-16 (2014)→ 138 Million parameters!!!!

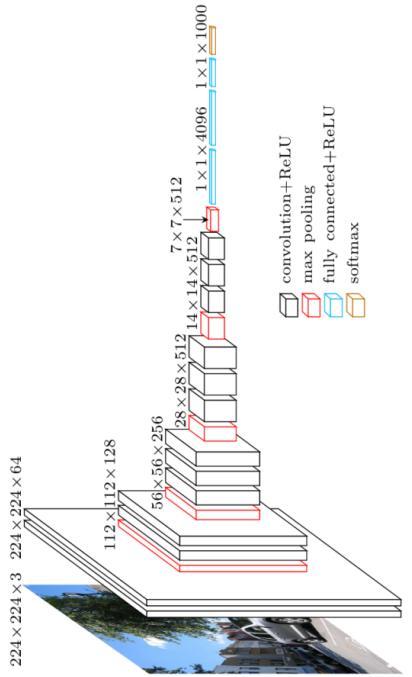


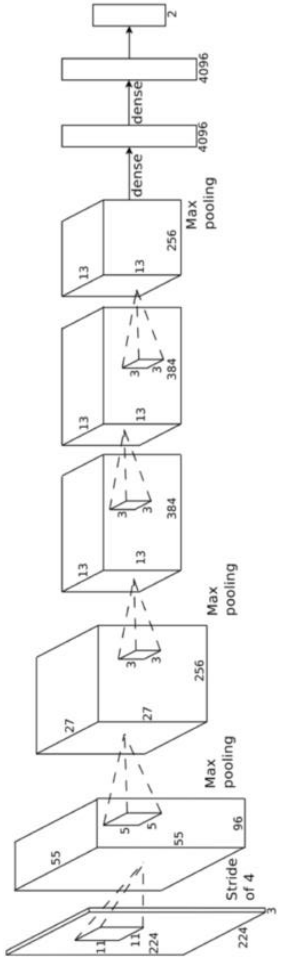
https://arxiv.org/abs/1409.1556

What can we do with such BFT (Best-Fitted and Trained?) models today?

Clustering!!!!! They do know how to extract features.
Unsupervised learning
Transfer Learning

Ben Kenobi - Old and Wise
lots of optimized parameters





Centro Brasileiro de Pesquisas Físicas

Redes Neurais profundas e aplicações
Deep Learning

Clécio Roque De Bom – debom@cbpf.br

clearnightsrthebest.com

CBPF