[M2, MVA]

Deep Learning

Maha ELBAYAD
maha.elbayad@student.ecp.fr

Homework 3

January 11, 2016

1 Theory

Dropout for linear regression:
Let's prove that the two following optimization problems are equivalent

$$\underset{w}{\text{minimize}}\ \mathbb{E}_{R\sim Bernoulli(p)}\left[\|y-(R\odot X)w\|^2\right] \tag{P1}$$

$$\underset{w}{\text{minimize}}\ \|y-pXw\|^2+p(1-p)\|\Gamma w\|^2,\ \Gamma=\left[diag(X^TX)\right]^{1/2} \tag{P2}$$

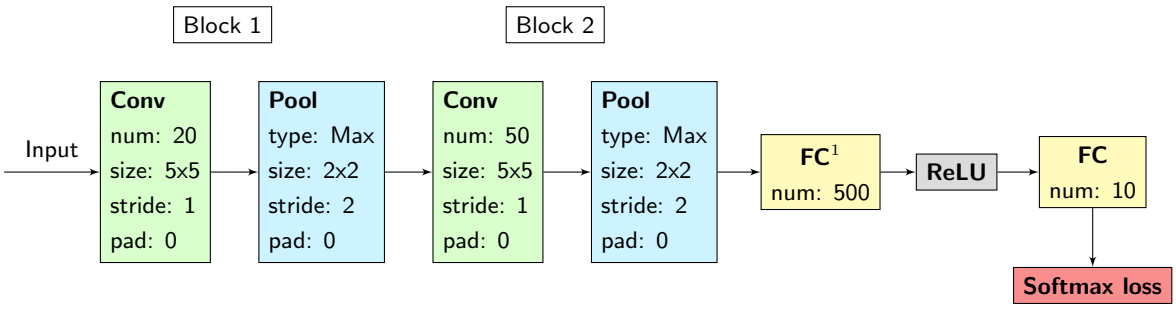
Where $R\in\{0,1\}^{N\times D}$ encodes the retained variables and $\odot$ denotes the Hadamard product.
We have:

$$\begin{aligned}
\|y-(R\odot X)w\|^2 &= y^Ty-2y^T(R\odot X)w+w^T(R\odot X)^T(R\odot X)w\\
\mathbb{E}_{R\sim Bernoulli(p)}[...] &= y^Ty-2py^TXw+w^T(var(R\odot X)+p^2X^TX)w \qquad (\mathbb{E}_R[R\odot X]=pX)\\
&= y^Ty-2py^TXw+w^T(p(1-p)diag(X^TX)+p^2X^TX)w\\
&= (y-pXw)^T(y-pXw)+p(1-p)w^T(diag(X^TX))w\\
&= \|y-pXw\|^2+p(1-p)\|\Gamma w\|^2 \quad \square
\end{aligned}$$

2 LeNet:

2.1 Experimentation (`MatConvNet v1.0-beta16`)

(V1 : baseline) We train a simplified version of LeCun's LeNet on a subset of the MNIST database. We synthetize the network's architecture in the following diagramm:



We train the network in 100 epoches and get the results shown in figure (1)

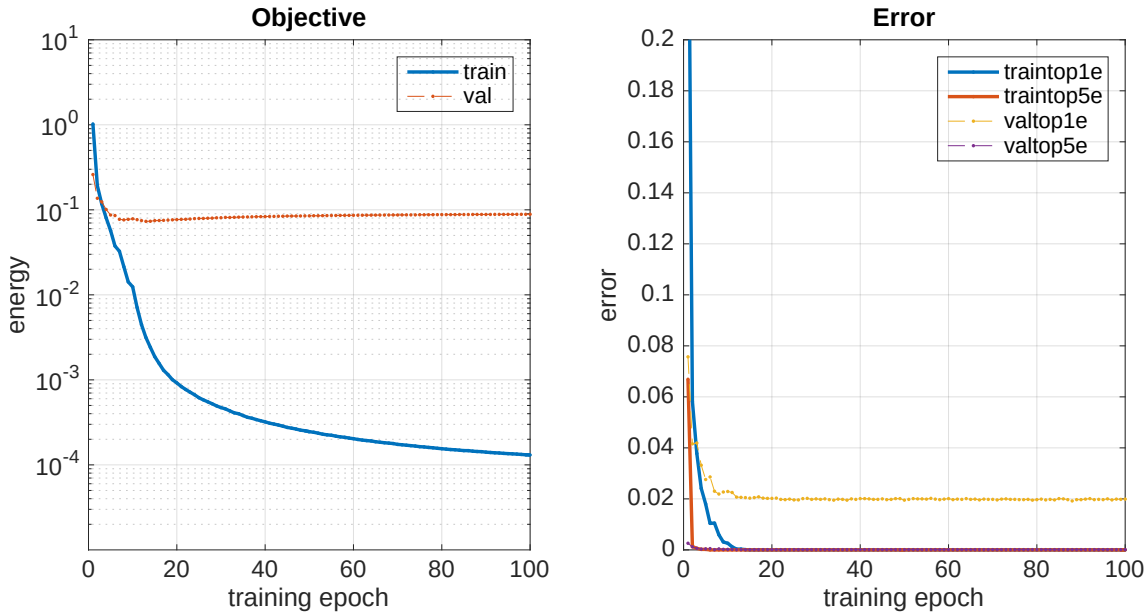
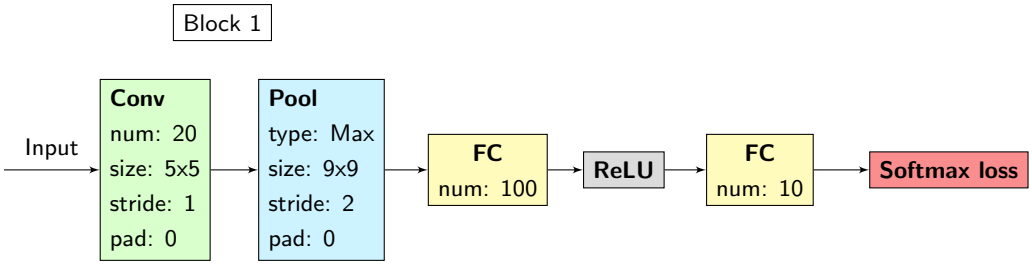


Figure 1: Baseline (2-Blocks) network

(V2) Reduce the depth of the net : A single block {`conv,pool`} - modified the pool layer to reduce the number of the network'parameters:



[1]FC for the fully connected layers

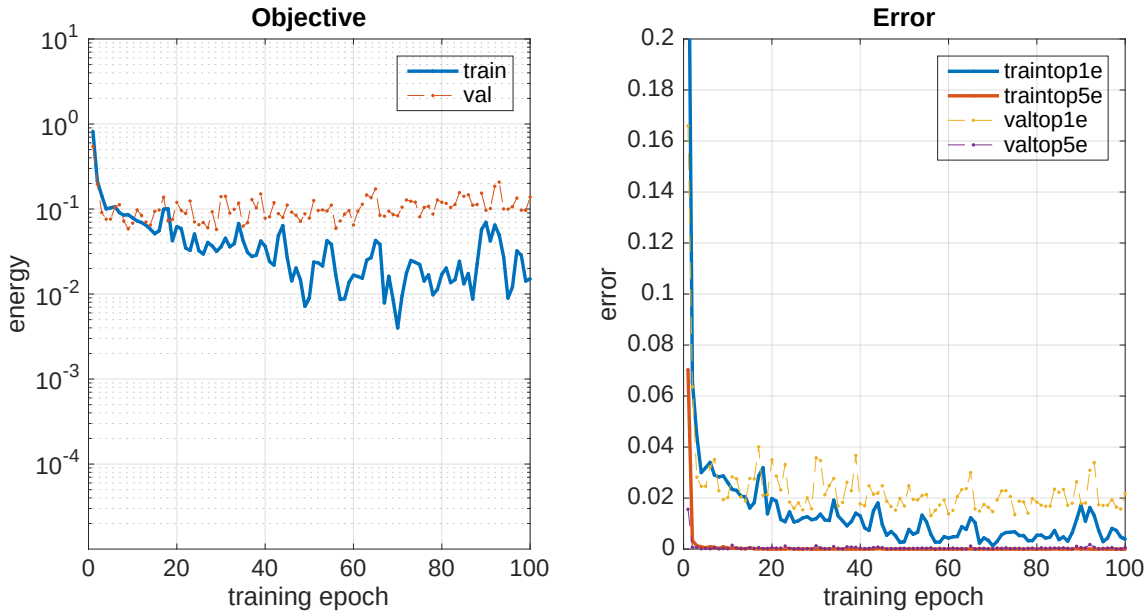


Figure 2: 1-Block network

(V3) Add an extra block {`conv,pool`}:

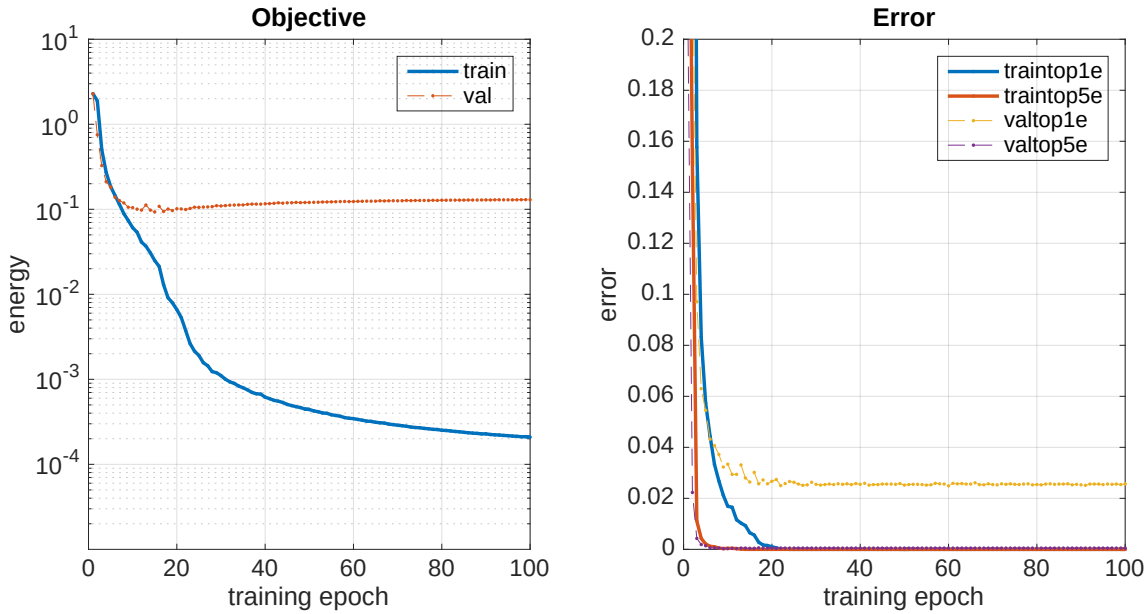
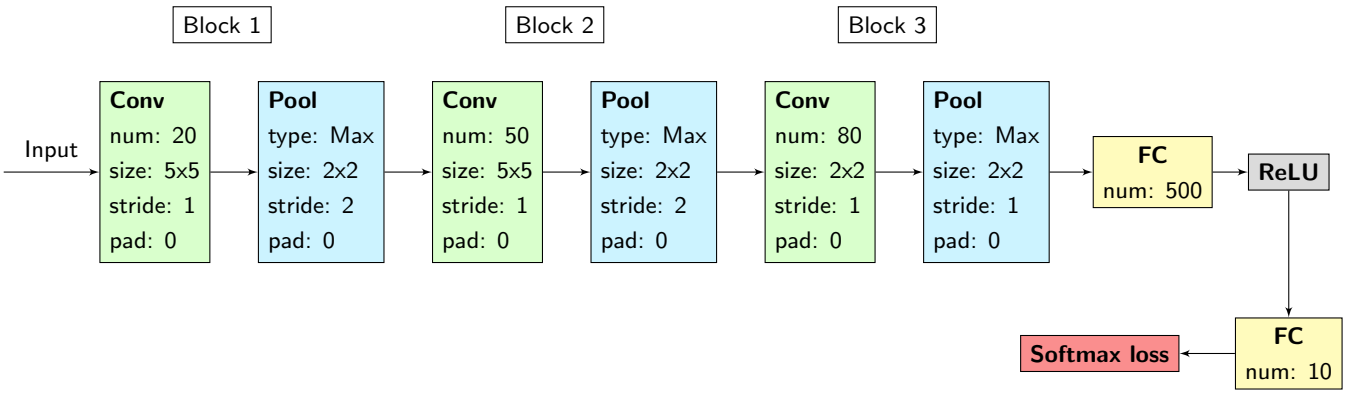
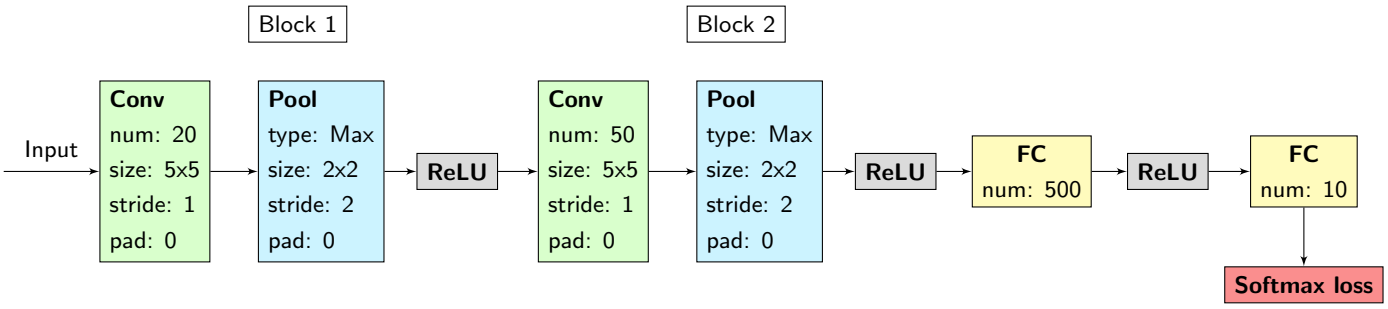


Figure 3: 3-Blocks network



(V4) 2 Blocks {`conv,pool,relu`}:



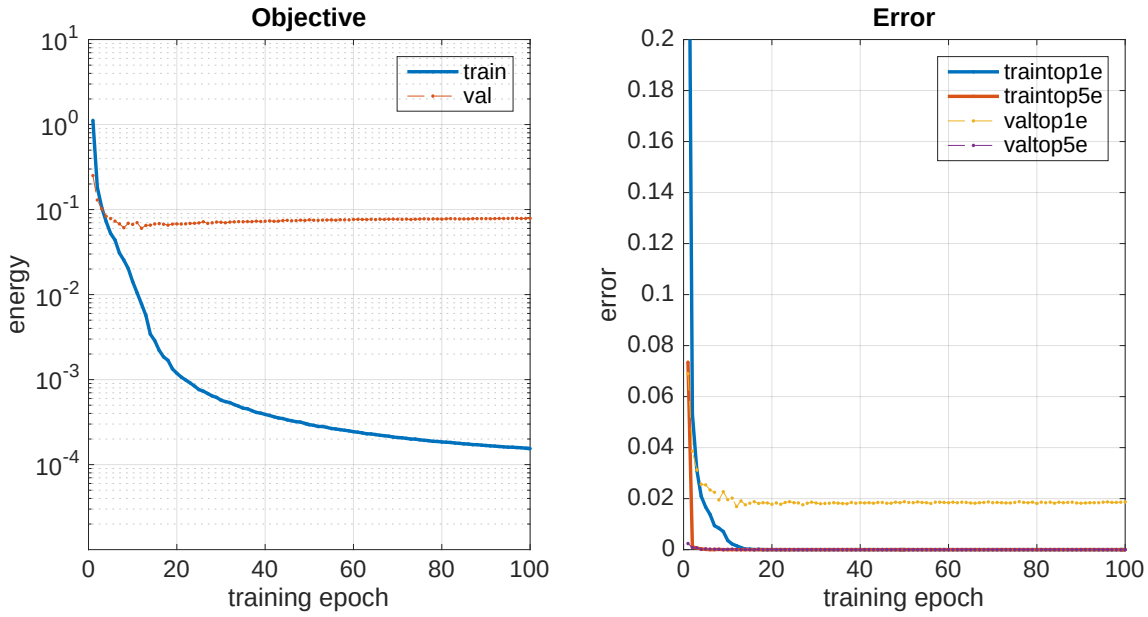


Figure 4: 2-Blocks network - {conv,pool relu}

(V5) Baseline with dropout (rate=.5) after the ReLU.

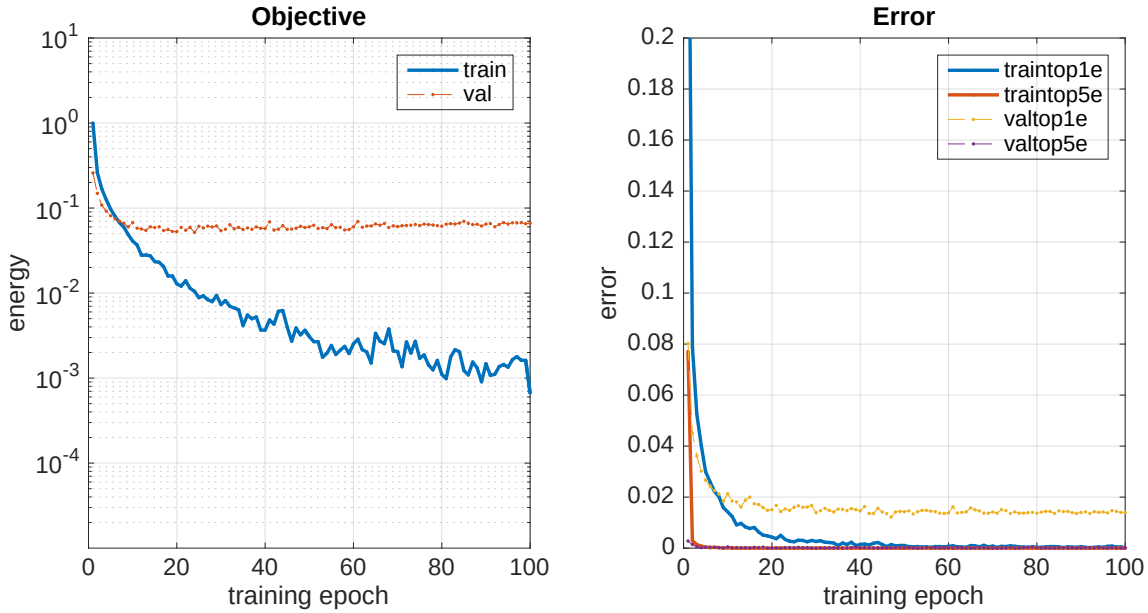


Figure 5: Baseline network - Added dropout

Maxout pooling implementation:

The output of the layer preceding the maxout is:

$$z = x^T \mathbf{W} + b \in \mathbb{R}^{H \times W \times K \times N}$$

Where:

- $(H \times W)$ size of the feature map.
- K: number of channels.
- N the batch size.
- $\mathbf{W}, b$ weights and biases of the preceding layer.

Forward - Channels pooling $(|groups| = G)$

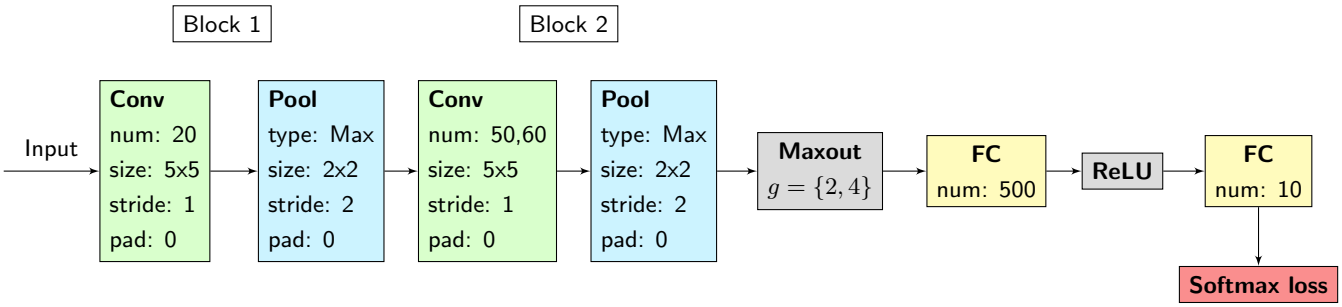
$$y_i = \max_{j \in K_i} z_{..j.},\ i \in 1, ..G,\ K_i \text{ indices of the } i^{th} \text{ group },\ y \in \mathbb{R}^{H \times W \times G \times N}$$

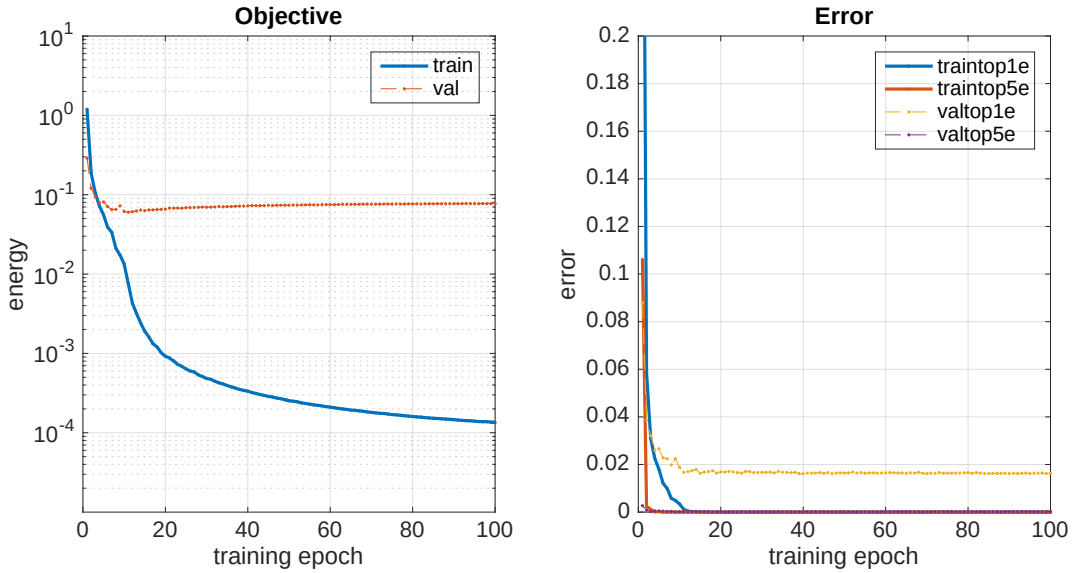
`Auxiliary output Mask:` boolean indicator of the filtered values ($\texttt{Mask} \in \mathbb{R}^{H \times W \times K \times N}$)

Backward Given $\frac{dz}{dy} \in \mathbb{R}^{H \times W \times G \times N}$,

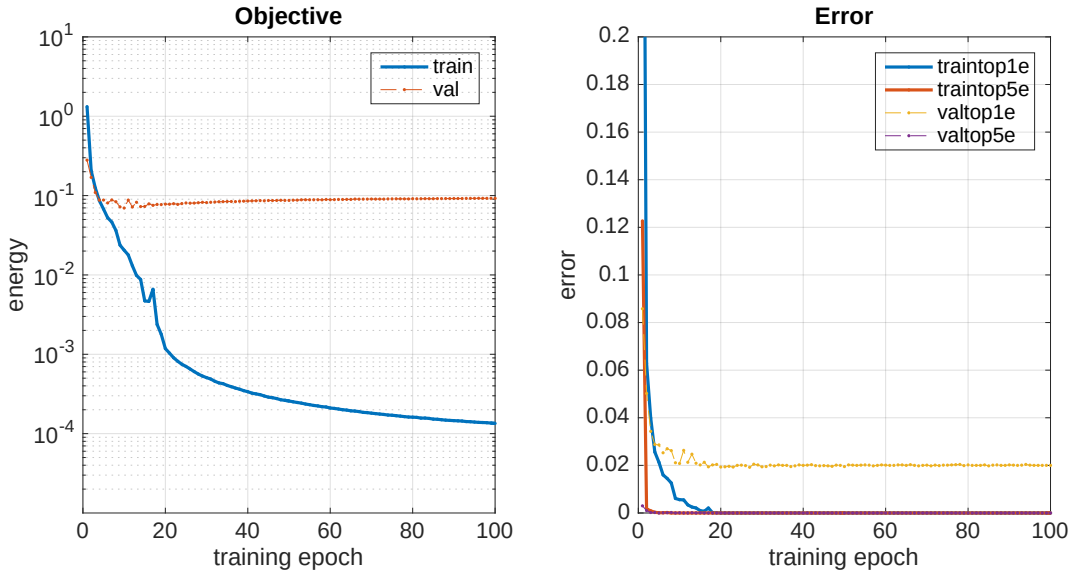
$$\frac{dz}{dx} = \underset{K_1, ...K_G}{\text{Duplicate along the channels dimension}} \frac{dz}{dy} \odot \texttt{Mask} \in \mathbb{R}^{H \times W \times K \times N}$$

(V6, 6bis) After implementing the maxout layer, we add it to th network after the 2nd block.





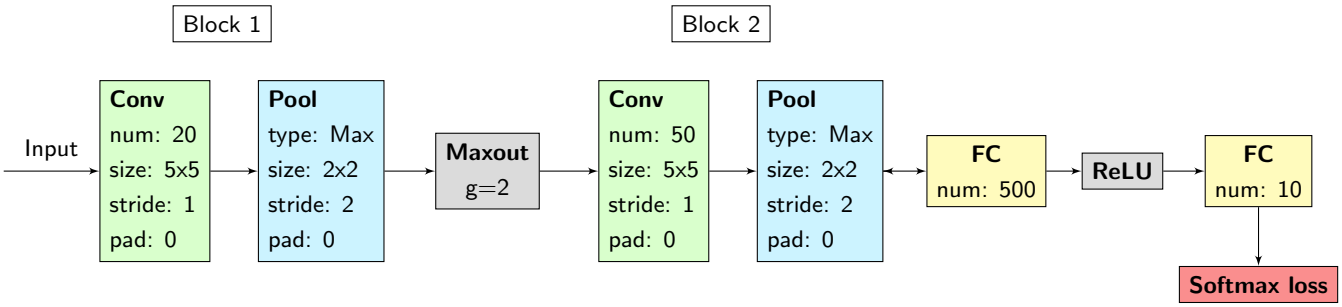
(a) g=2



(b) g=4

Figure 6: Baseline network - Maxout @Block2

(V7) Now we add a maxout layer to th network after the 1st block (baseline model)



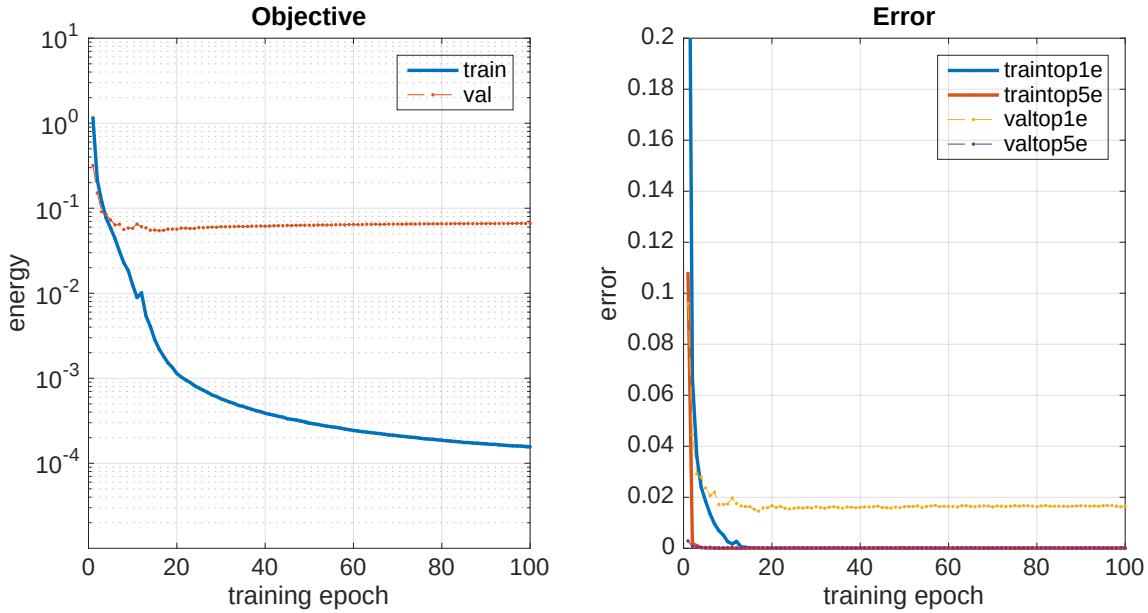
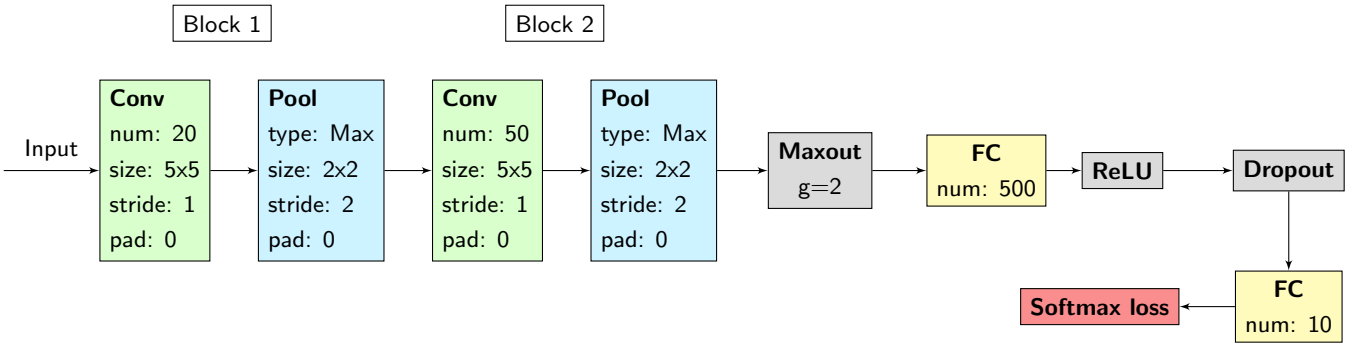


Figure 7: Baseline network - Maxout @Block1

(V8) We combine both the maxout after the 2nd block and the dropout after the ReLU.



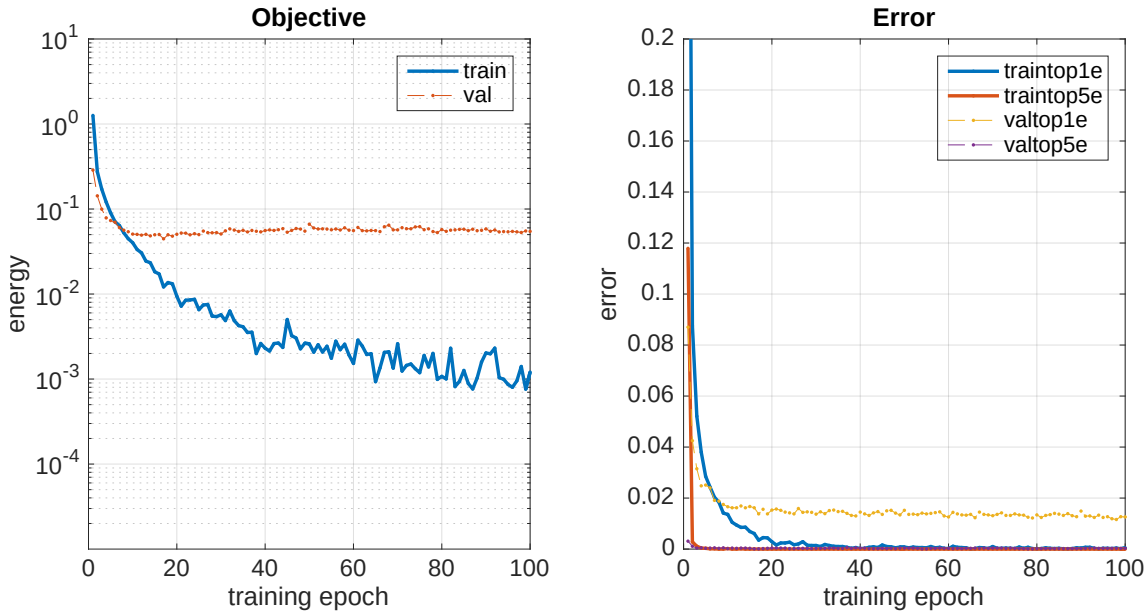


Figure 8: Baseline network - Maxout @Block1 + Dropout

(V9) We combine dropout with blocks `(Conv,Pool,Maxout)`

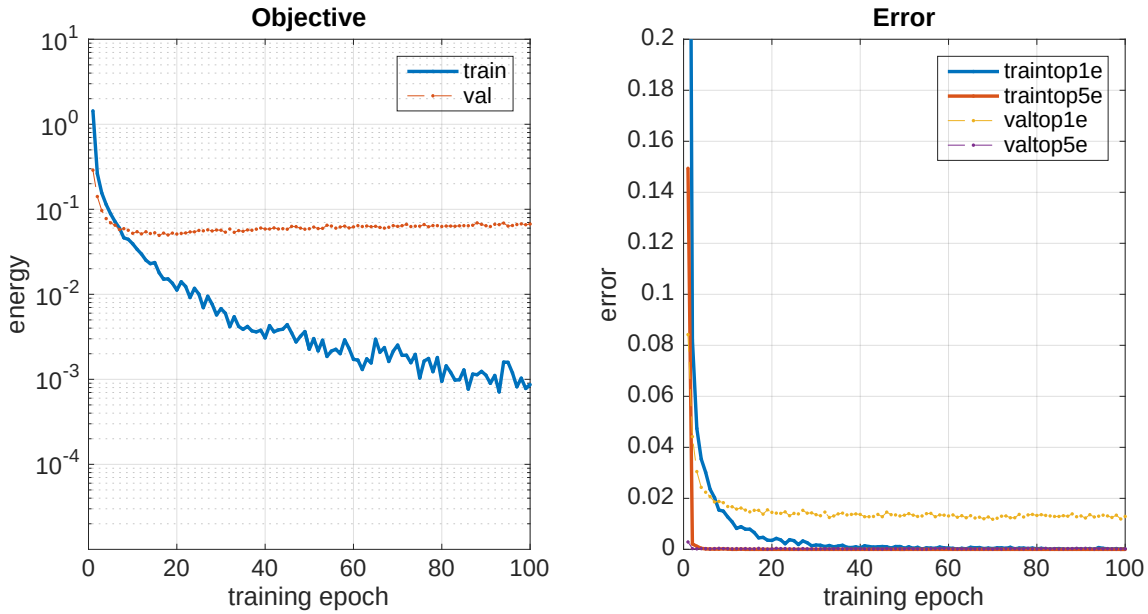
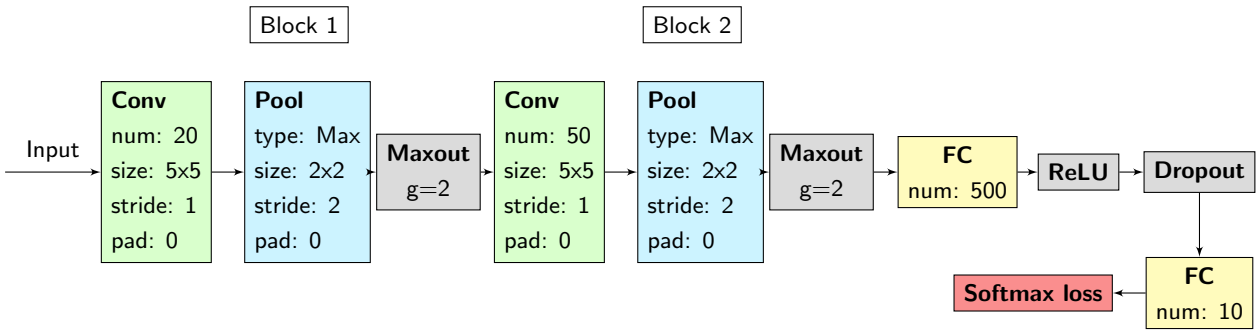


Figure 9: Baseline network - Maxout @Block1,2 + Dropout



(V10) $\sim$ (V9) with ReLU instead of Maxout.

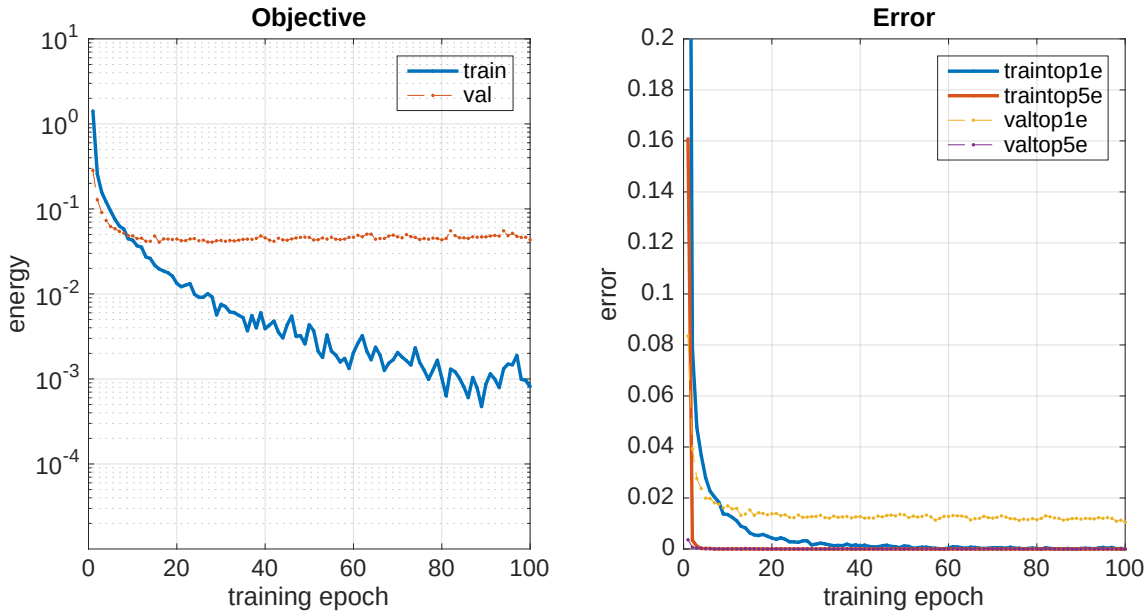
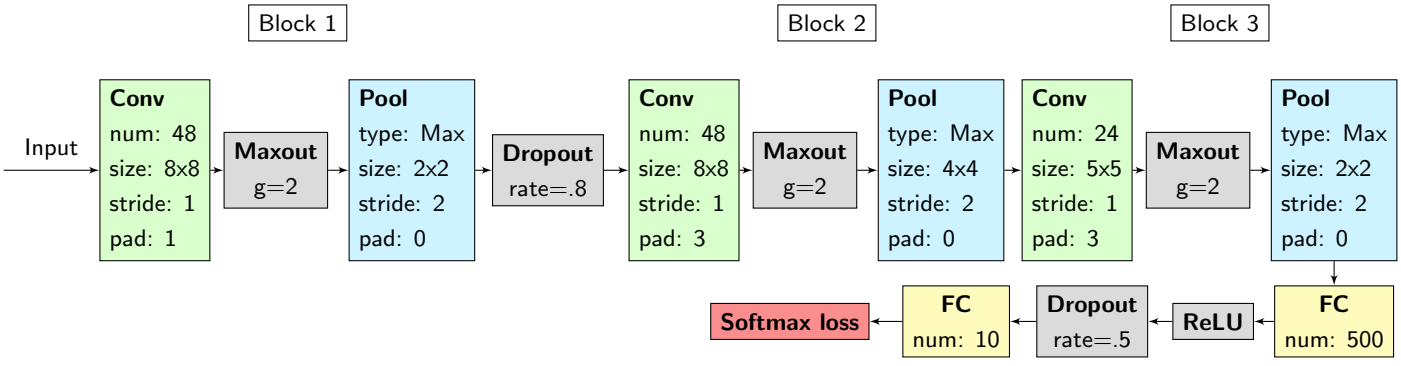


Figure 10: $2 \times$ (`Conv, Pool, ReLU`)+ Dropout

(V11) The model of (Goodfellow et al, 2013)



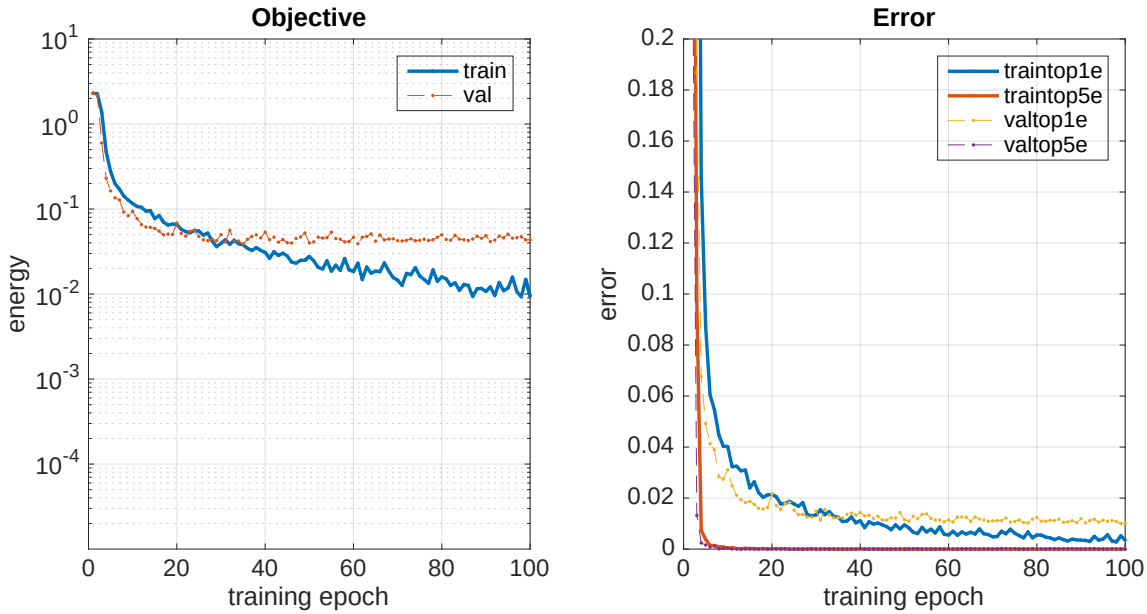


Figure 11: Goodfellow 2013

2.2 Summary:

Model	Validation Error	E(val)	E(train)[2]
v1 : 2 × (`Conv, Pool`)	1.99%	8.86e-02	1.31e-04
v2 : 1 × (`Conv, Pool`)	2.18%	1.38e-01	1.51e-02
v3 : 3 × (`Conv, Pool`)	2.57%	1.30e-01	2.09e-04
v4 : 2 × (`Conv, Pool, ReLU`)	1.87%	7.90e-02	1.54e-04
v5 : v1 + dropout	1.40%	6.62e-02	6.67e-04
v6 : v1 + maxout @block2 (g=2)	1.63%	7.74e-02	1.35e-04
v6 bis : v1 + maxout @block2 (g=4)	2.00%	9.23e-02	1.35e-04
v7 : v1 + maxout @block1	1.65%	6.63e-02	1.56e-04
v8 : v6 + dropout	1.26%	5.46e-02	1.21e-03
v9 : 2 × (`Conv, Pool, Max`) + dropout	1.29%	6.71e-02	8.67e-04
v10 : 2 × (`Conv, Pool, ReLU`) + dropout	**1.05%**	4.34e-02	8.16e-04
v11 : Goodfellow	**1.02%**	4.35e-02	9.41e-03

[2]Objective function at epoch 100

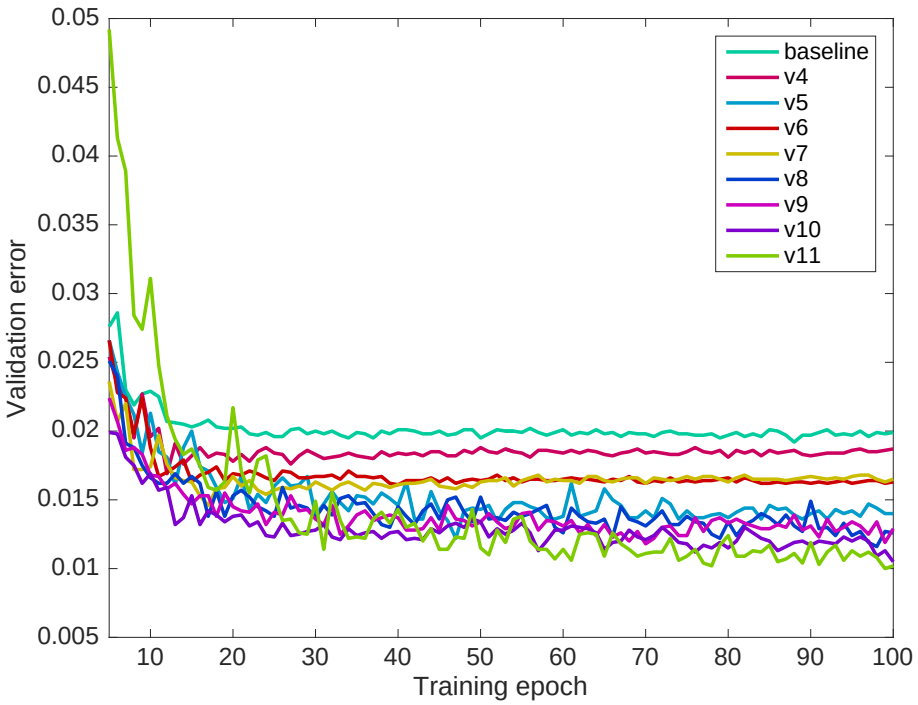


Figure 12

- (v1,v2,v3): The optimal depth of the LeNet network on the MNIST subset is of 2 blocks (performance degrading with depth>2).
- (v4): Adding ReLUs after pooling layers sparsifies the model and improves the results significantly.
- (v5) Adding dropout sparsifies the network responses and boosts its performance. And with its regularization effect, we avoid overfitting the training set.
- (v6, v6bis, v7): adding a maxout layer generally improves the results. With g=2 the model's complexity is higher, and so is its performance.
- (v8,v9): combining maxout and dropout (@ the 1^{st} FC) ensures the sparsity of the responses without dead units at the convolutional part of the network (Although the final test error is in favor of v8, the evolution throughout the training shows that the two models are quite equivalent).
- (v10): The model with ReLUs outperforms the maxout network.
- (v11): We use larger convolution filters with an extra block and a different arrangement of layers, consequently the performance of the maxout network is enhanced. (although the filters are large, the extra dropout sparsifies the model)