Lab CudaVision
Learning Vision Systems on Graphics Cards (MA-INF 4308)

Optimization and Learning

18.05.2021

PROF. SVEN BEHNKE, ANGEL VILLAR-CORRALES

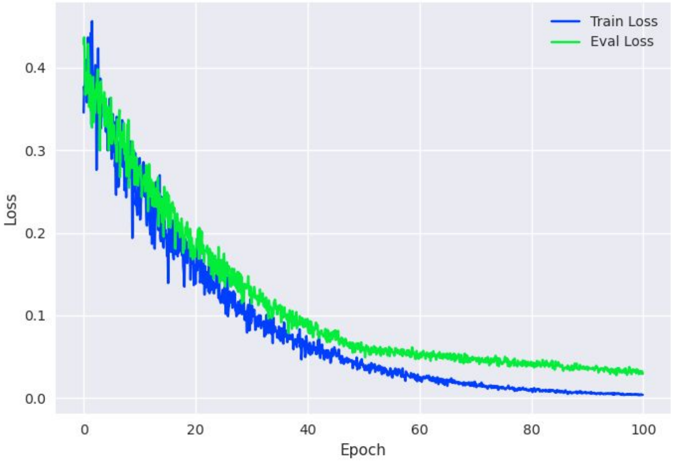
Contact: villar@ais.uni-bonn.de

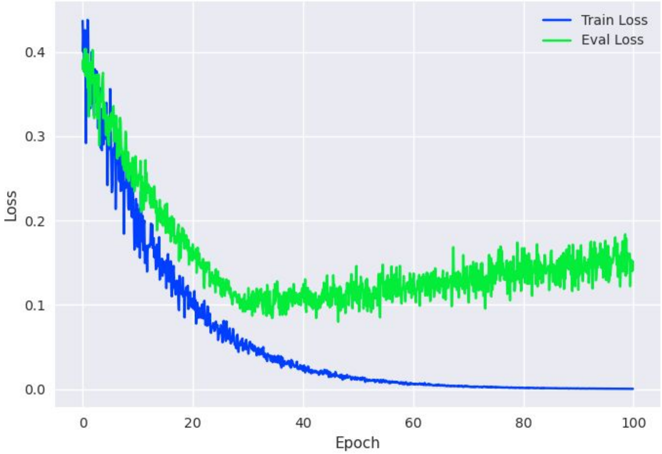
Understanding Learning Curves

Training and Evaluation Curves

- Give us information about how the model performs
  - Learning from training set
  - Generalization on validation set

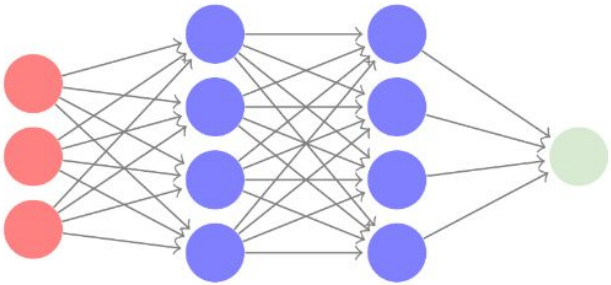
However...





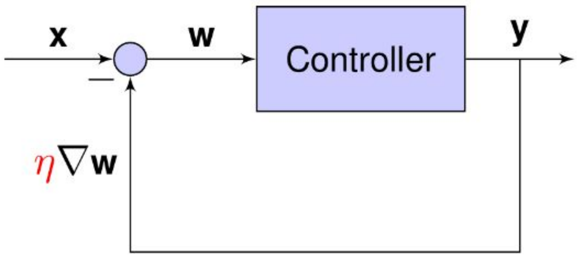
Stability in Neural Networks

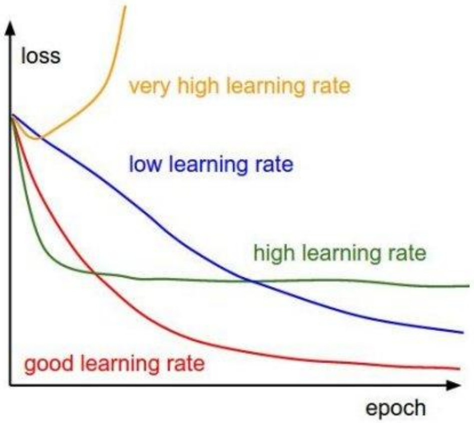
- What do these two images have in common?



- Both suffer from positive feedback
- Wrong choice of hyper-parameters can lead to disaster

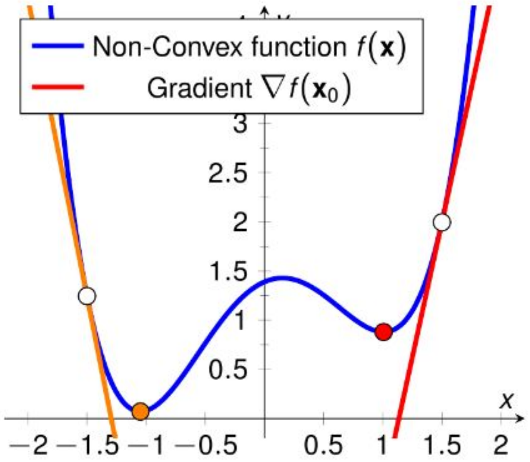
Effect of Learning Rate

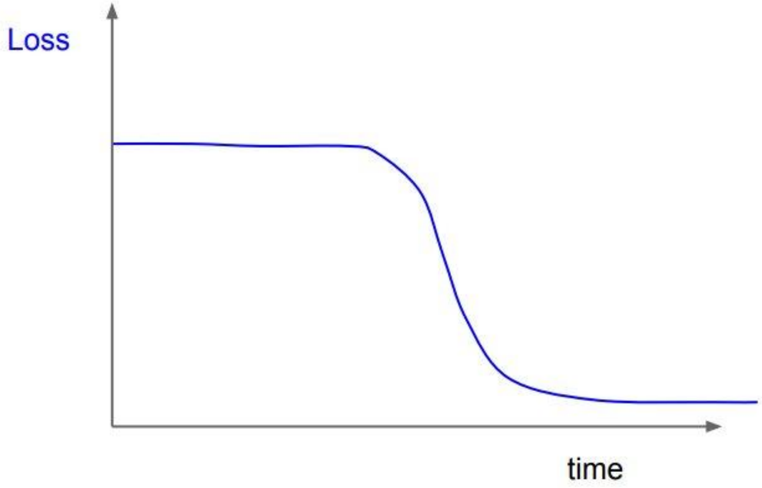




- η too high □ positive feedback □ loss stagnates or even grows
- η too small □ negative feedback □ loss decreases slowly or stagnates
- Choice of η is **critical** for learning!

Importance of Initialization

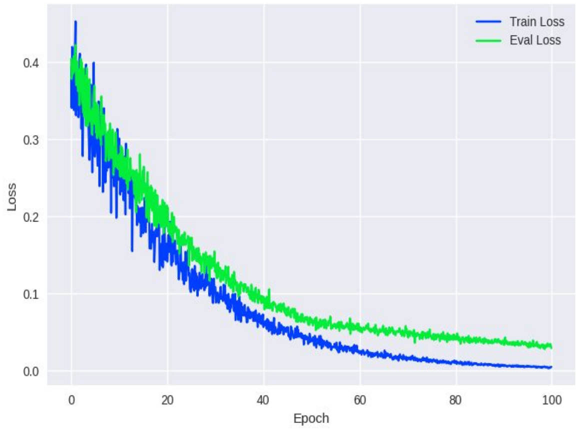




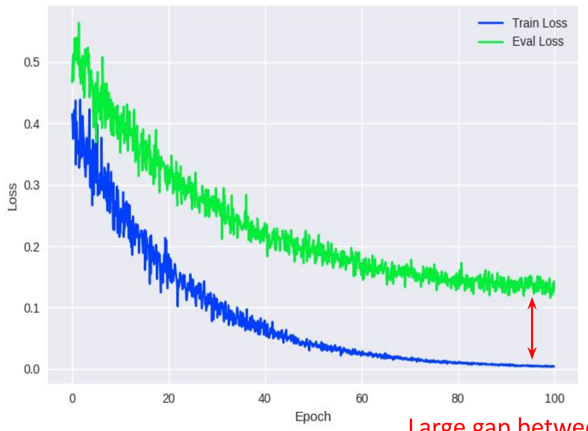
- Use a suitable initialization: Xavier or He
- Use prior knowledge for initialization

Diagnosing Model Behavior

Successful Training

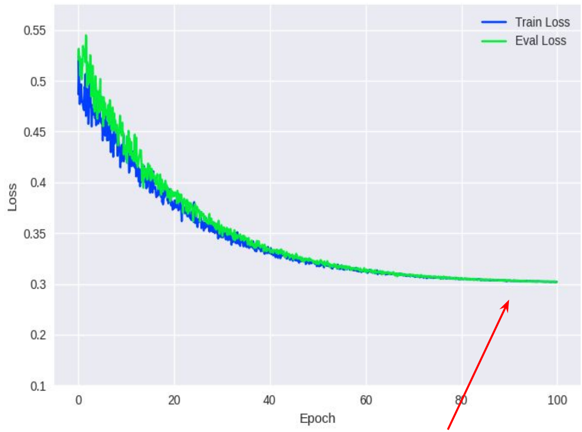


Overfitting



Large gap between train and eval loss

Underfitting



High loss values and no gap between train and eval loss

How to Avoid Underfitting?

- Add more layers to your model
- More powerful architectures designs
  - Residual connections
  - Dense connections
- Clean your data
  - Outliers
  - Noisy labels

How to Avoid Overfitting

- Reduce model capacity

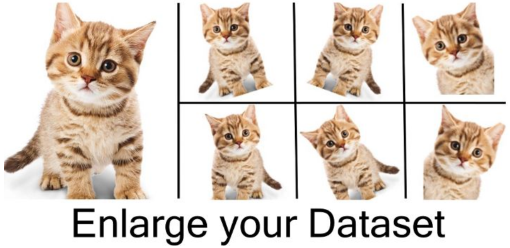
- Use more training examples
  - Gather more data
  - Data augmentation

- Regularization
  - Dropout
  - Weight regularization
  - Early stopping
  - Normalization (e.g. Batch Norm)

Regularization

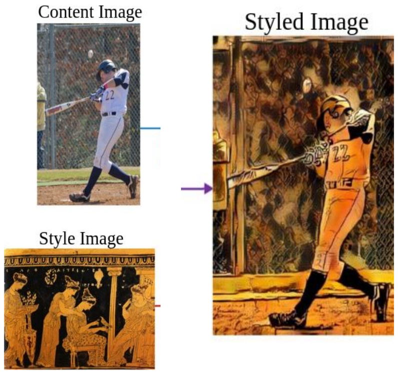
Data Augmentation

- Artificially enlarging your dataset
- Spatial or pixel transformations



Be careful!

- Generative models



Weight Regularization

- Enforcing certain priors on the model parameters
- Regularization applied to loss function

- **L2 Regularization (*Ridge* or weight decay)**
  - Enforce small parameter norm small parameters

$$\tilde{L}(\mathbf{w}, \mathbf{X}, \mathbf{Y}) = L(\mathbf{w}, \mathbf{X}, \mathbf{Y}) + \lambda \|\mathbf{w}\|_2^2$$

$$\mathbf{w}^{(k+1)} = \underbrace{(1 - \eta\lambda)\mathbf{w}^{(k)}}_{\text{Shrinkage}} - \eta \frac{\partial L}{\partial \mathbf{w}^{(k)}}$$

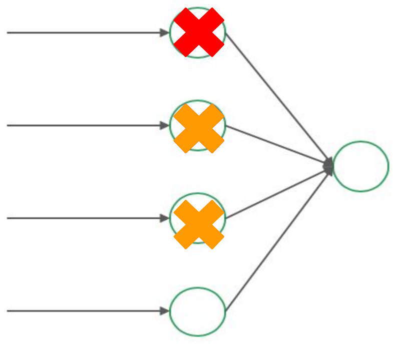
- **L1 Regularization (*Lasso*)**
  - Enforces sparsity in the parameters

$$\tilde{L}(\mathbf{w}, \mathbf{X}, \mathbf{Y}) = L(\mathbf{w}, \mathbf{X}, \mathbf{Y}) + \lambda \|\mathbf{w}\|_1$$

$$\mathbf{w}^{(k+1)} = \underbrace{\mathbf{w}^{(k)} - \eta\lambda \text{sign}\left(\mathbf{w}^{(k)}\right)}_{\text{Other shrinkage}} - \eta \frac{\partial L}{\partial \mathbf{w}^{(k)}}$$

- **L1 + L2 Regularization (*Elastic*)**

Dropout



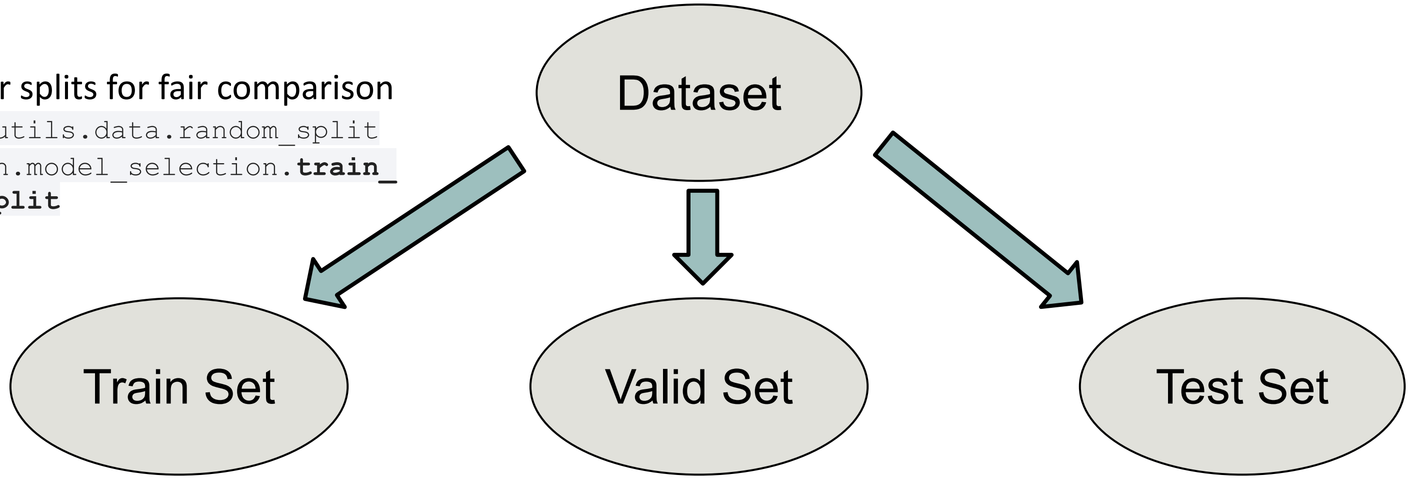
- During training, randomly set activations to zero with probability (*1-p*)
- At test time, we use all activations, but scaled by *p*
- Usual values
    - *p*=0.2 for input layer
    - *p*=0.5 for hidden layers

Normalizations

Dataset Splits

Splitting your Dataset

- Fix your splits for fair comparison
  - `Torch.utils.data.random_split`
  - `sklearn.model_selection.train_test_split`



- Train the model
- 60-70% of total

- Parameter Optimization
- Avoid overfitting
- 10-15% of dataset

- Evaluate performance
- Never use for training
- 15-20% of dataset

Generate your Dataset

- Annotated data is scarce and expensive to obtain
- Use synthetic data for training

Simple Generation

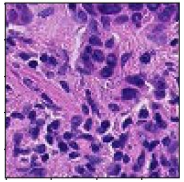
Simulators (CARLA)

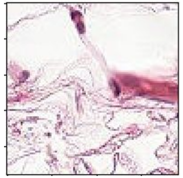
Game Engines (GTA V)

Evaluation Metrics

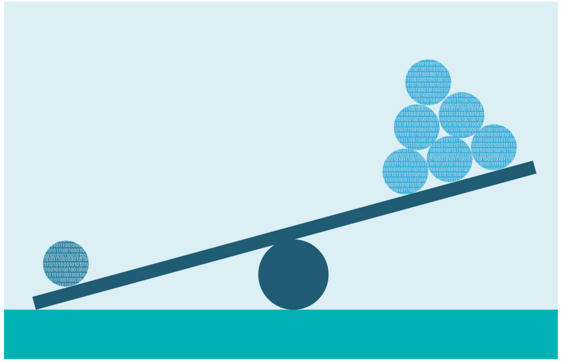
Problem with Accuracy

- "Let's image we are trying to detect if a person has some rare disease that is present in a 0.01% of the population based on some digital pathology scans"



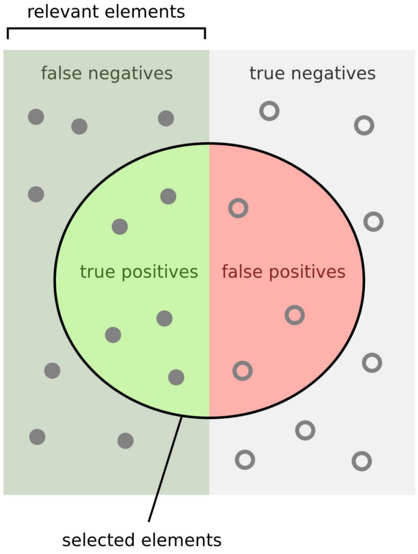


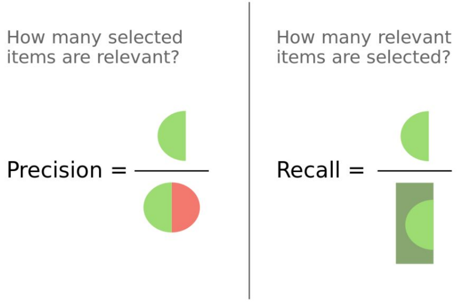
- Always predicting a negative results
  - Correct 99.99% of the time
- Accuracy is not always a good metric:
  - Imbalanced datasets
  - Not-permissible mistakes



Precision and Recall

- Precision, Recall and F1-Score are better suited metrics for many pattern recognition problems





$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$