Solving overfitting in neural networks

Valerio Velardo

Solving overfitting

- Simpler architecture
- Data augmentation
- Early stopping
- Dropout
- Regularization

Solving overfitting

- Simpler architecture
- Data augmentation
- Early stopping
- Dropout
- Regularization

Simpler architecture

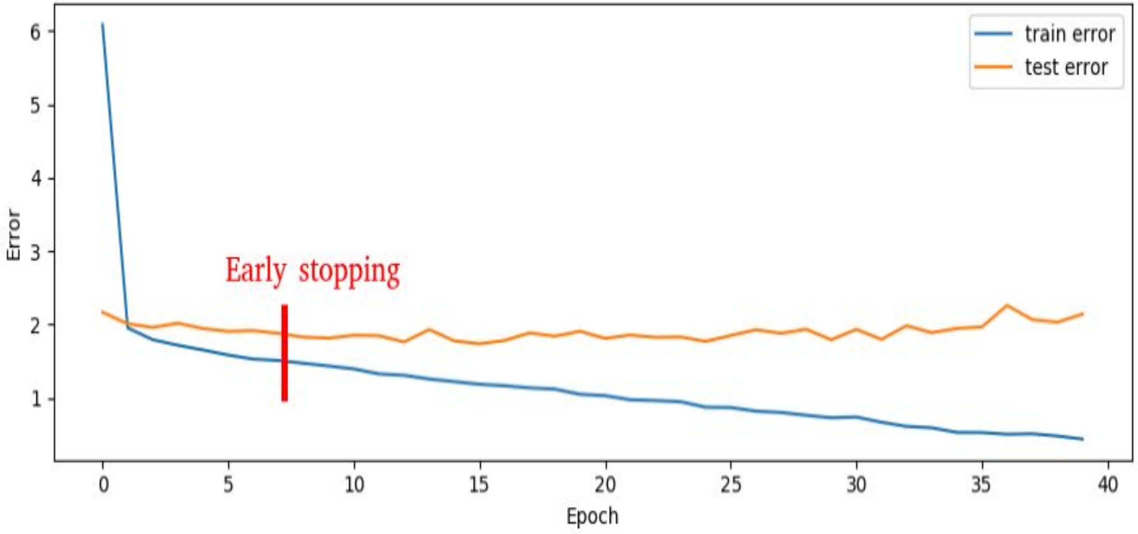
- Remove layers
- Decrease # of neurons
- No universal rule :(

Audio data augmentation

- Artificially increase # of training samples
- Apply transformations to audio files
    - Pitch shifting
    - Time stretching
    - Adding background noise
    - ...

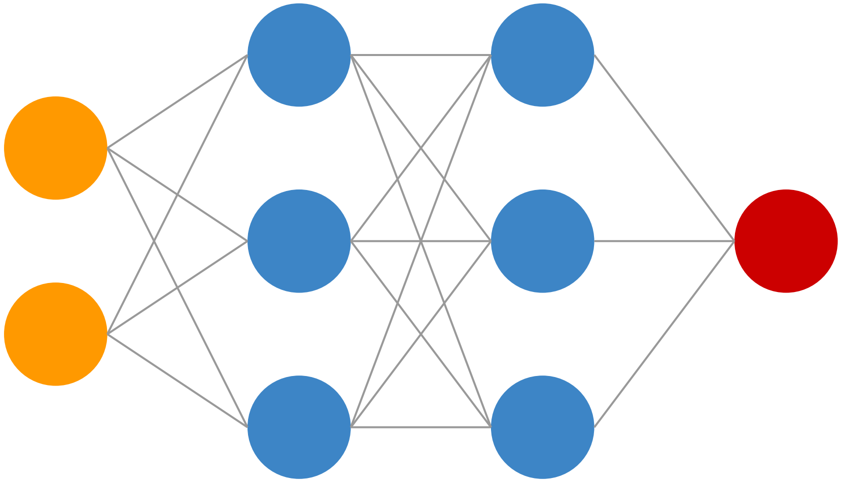
Early stopping

- Choose rules to stop training



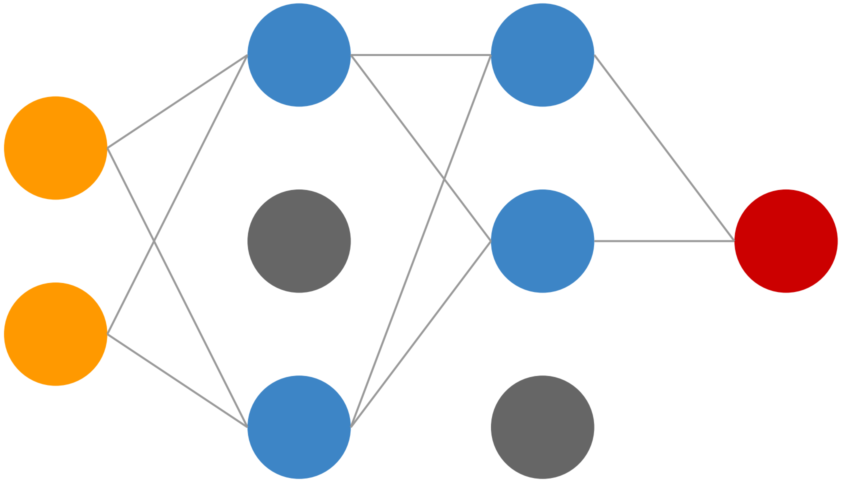
Dropout

- Randomly drop neurons while training
- Increased network robustness



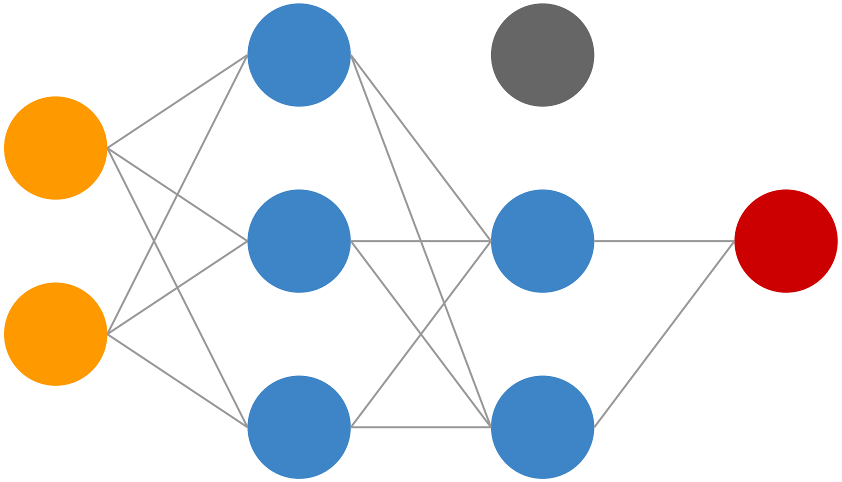
Dropout

- Randomly drop neurons while training
- Increased network robustness



Dropout

- Randomly drop neurons while training
- Increased network robustness



Dropout

- Randomly drop neurons while training
- Increased network robustness
- Dropout probability: 0.1-0.5

Regularization

- Add penalty to error function
- Punish large weights
- L1 and L2

L1 regularization

- Minimises absolute value of weights
- Robust to outliers
- Generates simple model

$$E(\mathbf{p}, \mathbf{y}) = \frac{1}{2}(\mathbf{p} - \mathbf{y})^2 + \boxed{\lambda \sum |W_i|}$$

L1 regularization

- Minimises absolute value of weights
- Robust to outliers
- Generates simple model

$$E(\mathbf{p}, \mathbf{y}) = \frac{1}{2}(\mathbf{p} - \mathbf{y})^2 + \boxed{\lambda} \sum |W_i|$$

L2 regularization

- Minimises squared value of weights
- Not robust to outliers
- Learns complex patterns

$$E(\mathbf{p}, \mathbf{y}) = \frac{1}{2}(\mathbf{p} - \mathbf{y})^2 + \boxed{\lambda \sum W_i^2}$$