Computational Analysis of Sound and Music

Deep Learning - Fundamentals

Dr.-Ing. Jakob Abeßer
Fraunhofer IDMT

jakob.abesser@idmt.fraunhofer.de

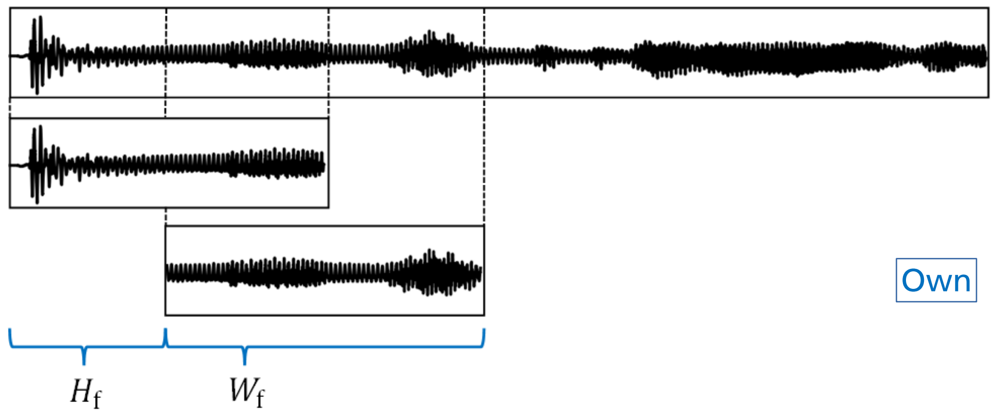
Outline

- **Audio Data Representations**
- Data Normalization
- Data Augmentation
- Deep Learning
- Multi-Layer Perceptron

Audio Data Representation

One-Dimensional Representations

- Audio samples **frames** (“end-to-end learning”)



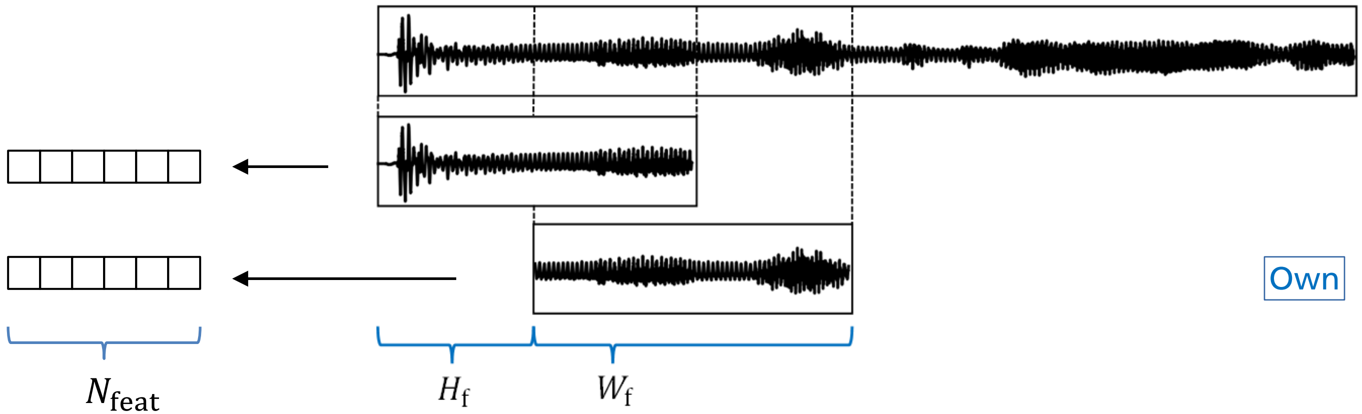
Own

  - Hopsize H_f
  - Blocksize W_f
- Number of frames $N_\mathrm{f} = \left\lfloor \frac{N - W_\mathrm{f}}{H_\mathrm{f}} \right\rfloor + 1$
- Tensor $\mathcal{X} \in \mathbb{R}^{N_\mathrm{f} \times W_\mathrm{f}}$

Audio Data Representation

One-Dimensional Representations

- Feature vectors from audio sample **frames**

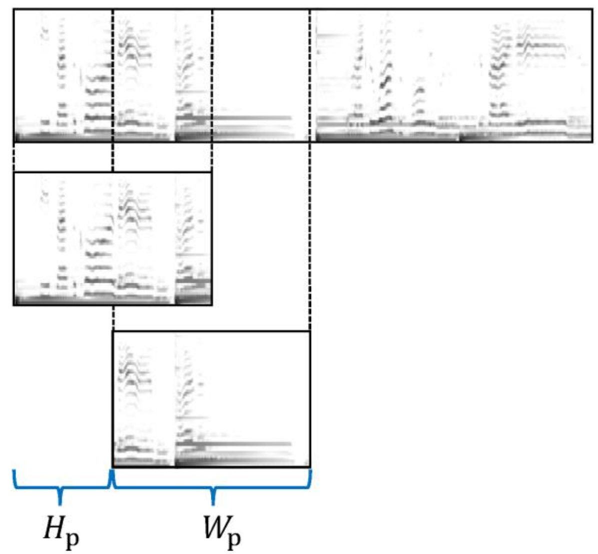


  - Example: MFCC features (N_{feat} = 13)
- Tensor $\mathcal{X} \in \mathbb{R}^{N_{\text{f}} \times N_{\text{feat}}}$

Audio Data Representation

Two-Dimensional Representations

- 2D representation
  - Spectrogram **patches**
    - "Image-like" input
    - Commonly for convolutional layers
  - Number of frequency bins F
  - Patch hopsize H_p
  - Window length W_p
- Number of patches $N_\mathrm{p} = \left\lfloor \frac{N_\mathrm{f} - W_\mathrm{p}}{H_\mathrm{p}} \right\rfloor + 1$
  - Tensor $\mathcal{X} \in \mathbb{R}^{N_\mathrm{p} \times F \times W_\mathrm{p}}$



Own

Outline

- Audio Data Representations
- **Data Normalization**
- Data Augmentation
- Deep Learning
- Multi-Layer Perceptron

Data Normalization

Why?

- Consistent scale of input data
    - Stabilizes learning
        - Equalizes feature contribution to learning algorithm (gradient descent)
    - Accelerates convergence
    - Prevents numerical instability
        - Reducing risk of exploding/vanishing gradients

Data Normalization

How?

- Statistics can be computed per
  - Patch / Batch (later …) / Frequency bin / Dataset
- **Min-Max Normalization**
  - Rescales input data to $[0, 1]$
  - Distribution outliers can cause skewed data distributions

$$x_i \leftarrow \frac{x_i - \min x}{\max x \ - \min x}$$

`Scikit-learn: sklearn.preprocessing.MinMaxScaler`

Data Normalization

How?

- **Standardization** (z-score normalization)
  - Modify data distribution to zero-mean and unit-variance
  - Distribution outliers can cause skewed data distributions

$$x_i \leftarrow \frac{x_i - \bar{x}}{\delta + \varepsilon}$$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

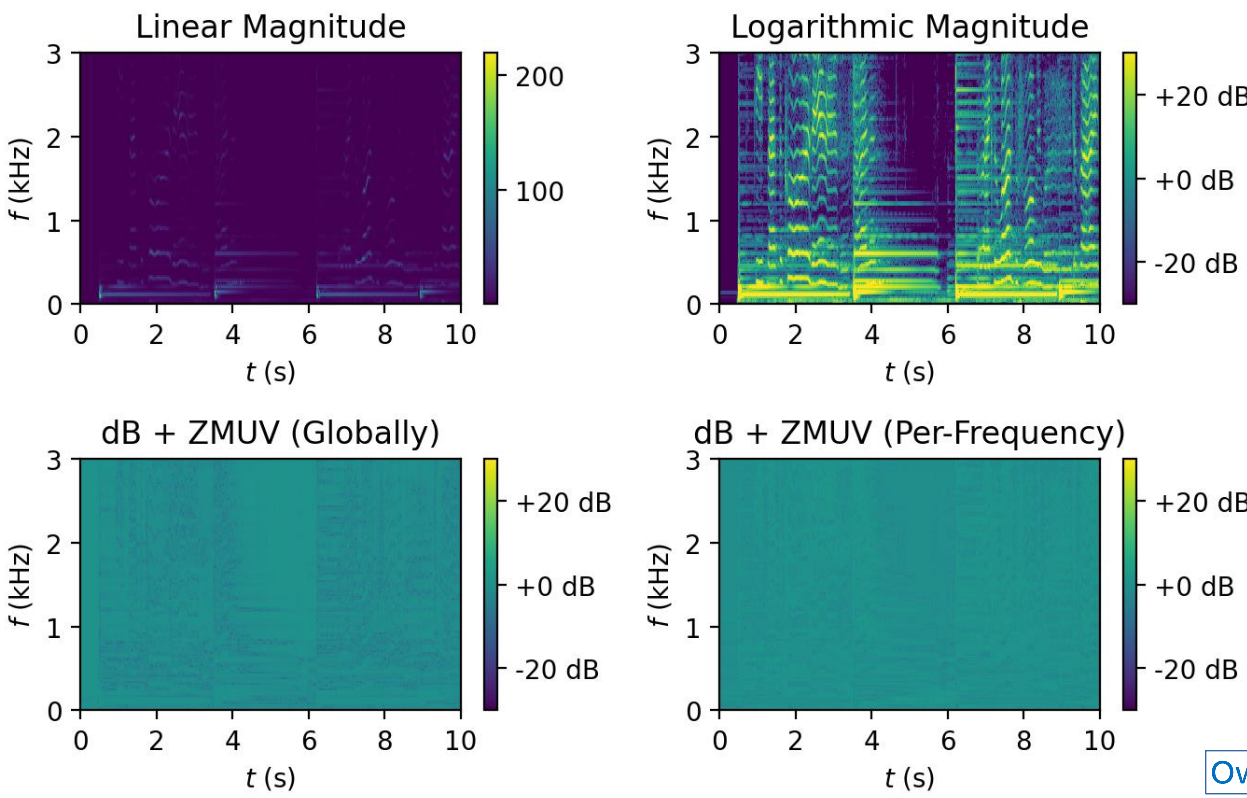
$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2$$

`Scikit-learn: sklearn.preprocessing.StandardScaler`

Data Normalization

How?

- Example



Own

Outline

- Audio Data Representations
- Data Normalization
- **Data Augmentation**
- Deep Learning
- Multi-Layer Perceptron

Data Augmentation

Why?

- Reduces data scarcity (limited amount of training data)
- Diversified training data
    - Enhanced generalization towards new, unseen data
    - Improved robustness towards changes in the input data

Data Augmentation

How?

- Audio Signal Processing
    - Time-Shift
    - Pitch-Shift
    - Noise
- Make sure not to alter semantic annotation

Audiomentations

Data Augmentation

How?

- Computer Vision Techniques
    - Sometimes effective (but lack intuition in audio domain)

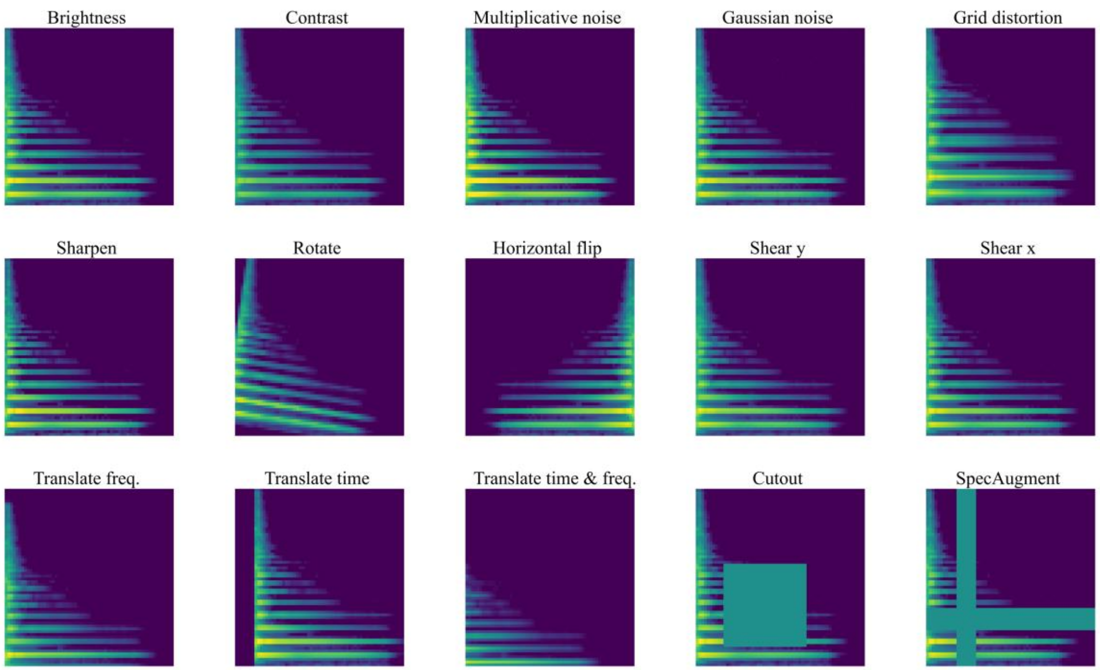


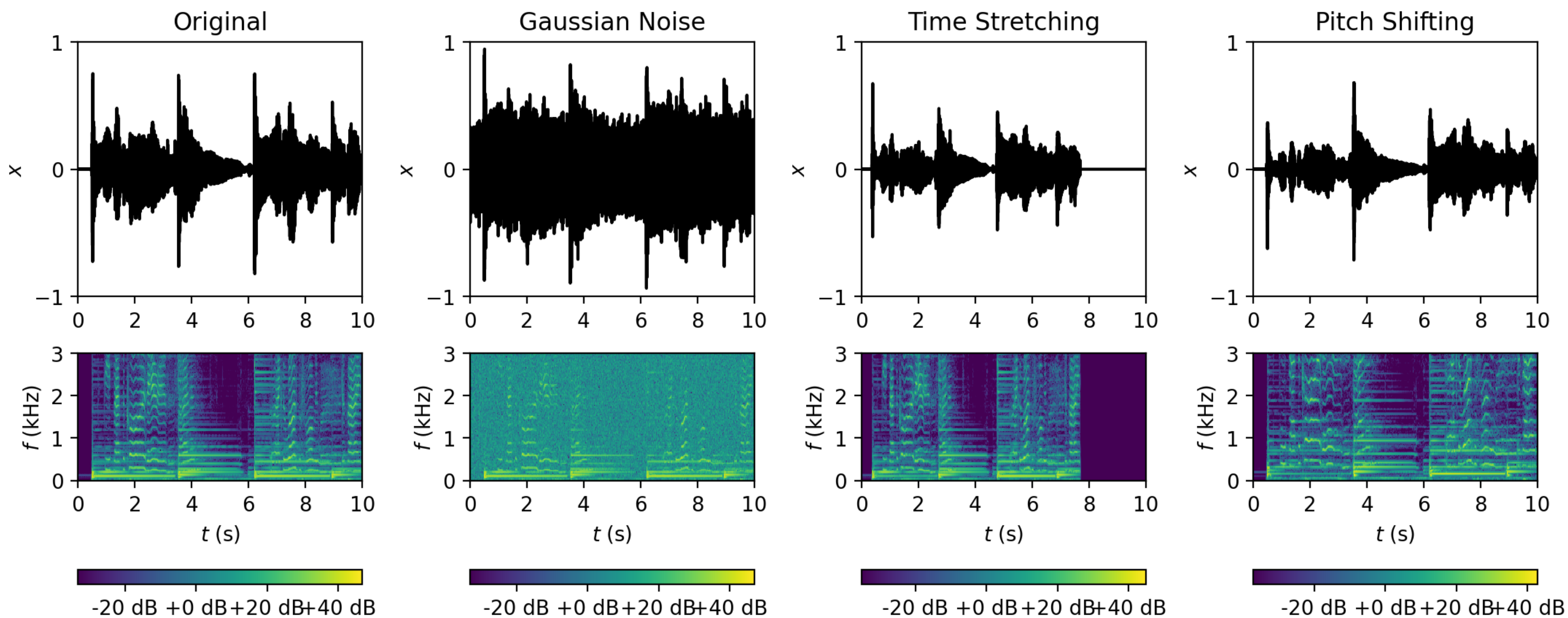
Fig-D2-1

Albumentations

Data Augmentation

How?

- Example



Own

Outline

- Audio Data Representations
- Data Normalization
- Data Augmentation
- **Deep Learning**
- Multi-Layer Perceptron

Deep Learning

Introduction

- Artificial neural networks → mimic brain processing
    - Connected neurons
    - Weighted input summation
    - Non-linear processing
- Shallow networks → **deep** networks

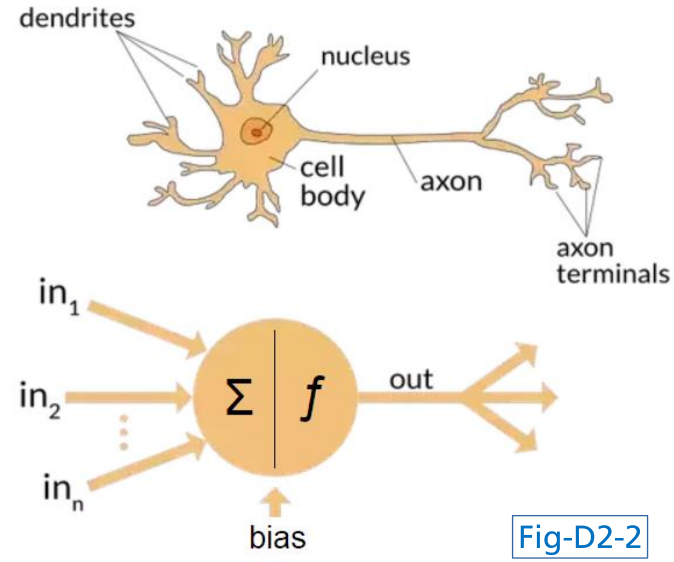


Fig-D2-2

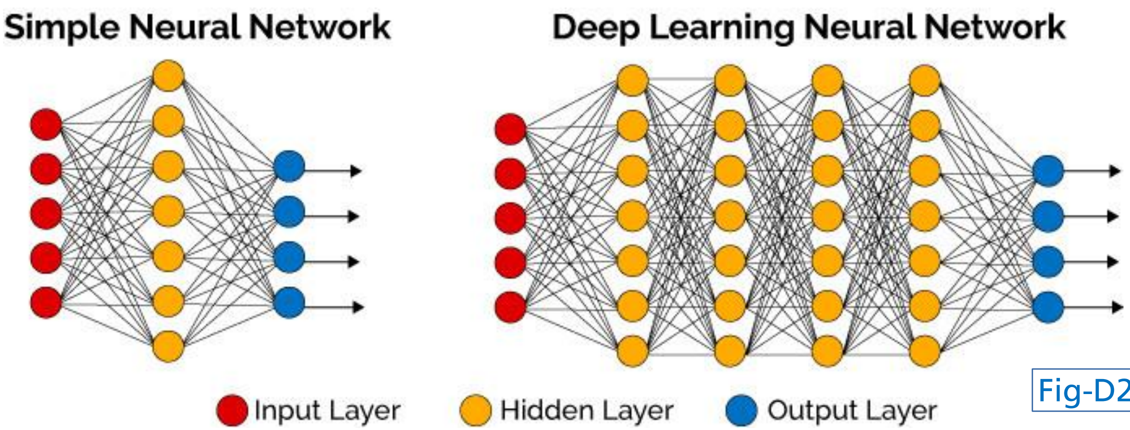


Fig-D2-3

Deep Learning

Hierarchical Feature Learning

- Example: Face recognition

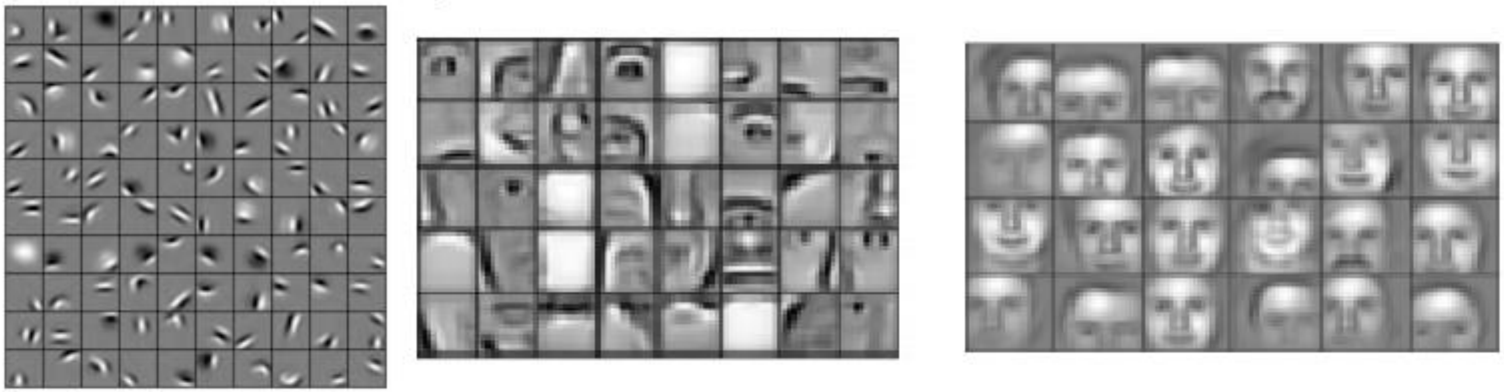


Fig-D2-4

Edges, curves | Shapes, object parts | Objects (faces)

First layers → Final layers

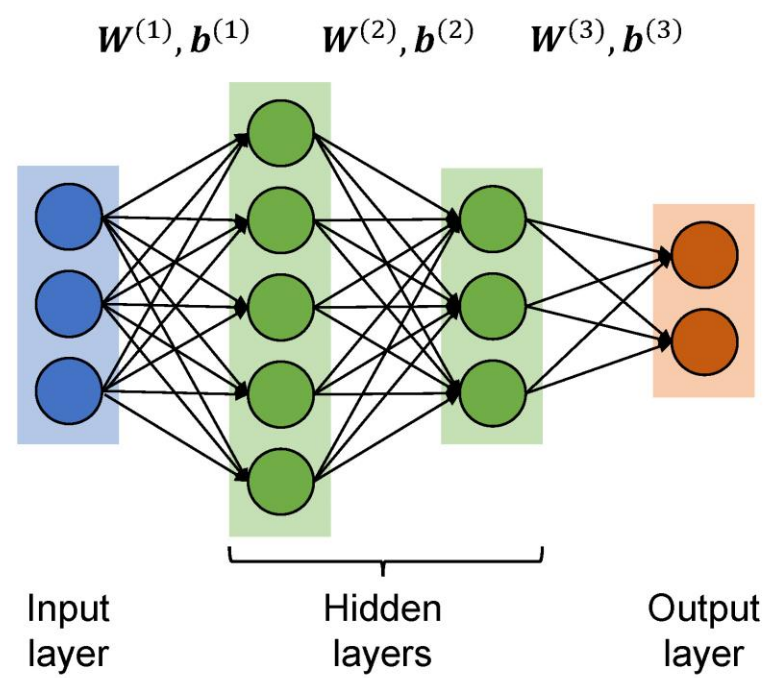
Outline

- Audio Data Representations
- Data Normalization
- Data Augmentation
- Deep Learning
- **Multi-Layer Perceptron**

Multilayer Perceptron (MLP)

Components

- Dense/Fully-connected neural network
- Trainable parameters
  - **Weights** W
  - **Biases** b
- Hyperparameters
  - Number of layers (depth)
  - Layer width



Own

`tensorflow.keras.layers.Dense`

Multilayer Perceptron (MLP)

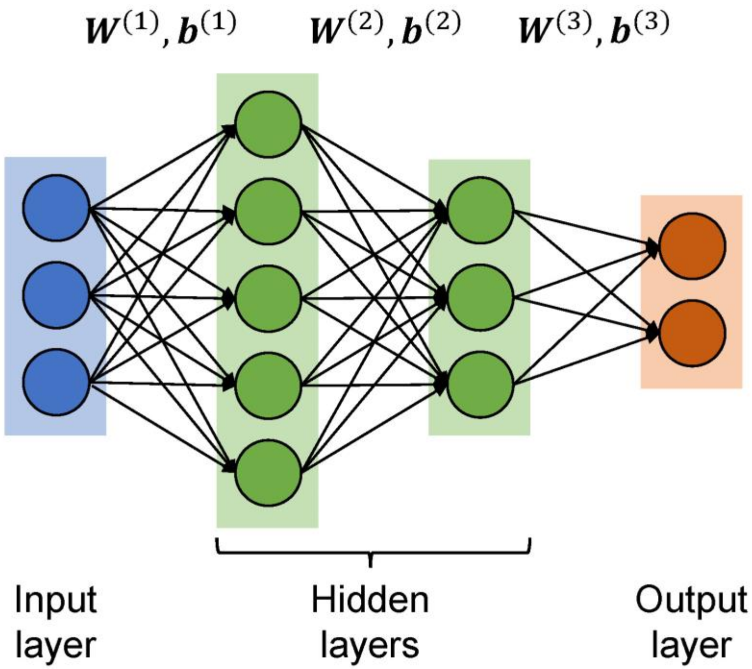
Operations

- Operations in layer l
  - Weighted input summation

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$

  - Non-linear activation function

$$a^{(l)} = f^{(l)}\left(z^{(l)}\right)$$



Own

`tensorflow.keras.layers.Dense`

Multilayer Perceptron (MLP)

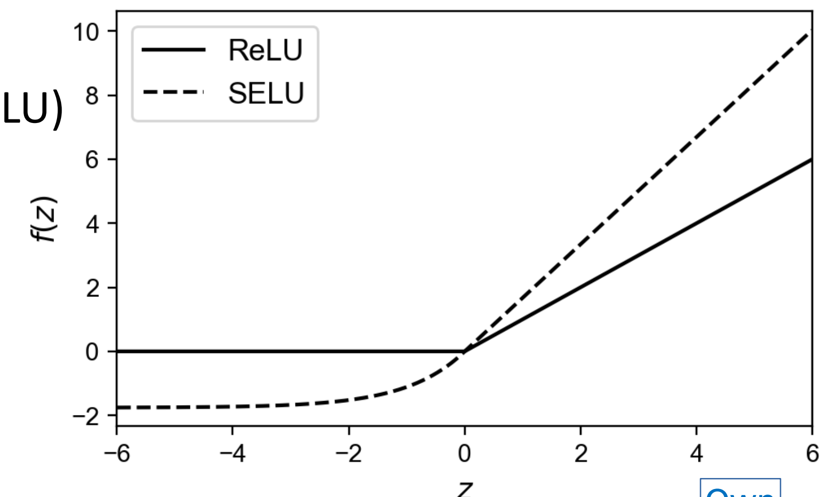
Activation Functions

- **Non-linear** operations allow DNNs (of limited depth) to model complex IO mappings
- Two examples
  - Rectified Linear Activation Function (ReLU)

$$f(z) = \max(0, z)$$

  - Scaled Exponential Linear Units (SELU)

$$f(z) = \begin{cases} \lambda z, & \text{if } z > 0 \\ \lambda\alpha(e^{z} - 1), & \text{otherwise} \end{cases}$$



Own

`tensorflow.keras.layers.Activation`

Multilayer Perceptron (MLP)

Activation Functions (Output Layer)

- Depends on modeling task
  - **Regression** → **Linear** AF

$$f(z) = z$$

  - **Multilabel classification** → **Sigmoid** AF (independent probability values for C classes: $f(z)\colon \mathbb{R}^C \to (0,1)^C$)

$$f(z)_c = \frac{1}{1 + e^{-z_c}}$$

  - **Multiclass classification** → **Softmax** AF (Probability distribution over C classes: $f(z)\colon \mathbb{R}^C \to (0,1)^C, \sum_{c=1}^{C} f(z)_c = 1$

$$f(z)_c = \frac{e^{z_c}}{\sum_{j=1}^{C} e^{-z_j}}$$

Multilayer Perceptron (MLP)

Model Training

- Training → Optimization problem

$$\hat{\theta} = \arg\min_{\theta} L(\theta)$$

  - θ → Model parameters
  - L → Loss function
- Gradient-based optimization

$$\theta \coloneqq \theta - \epsilon \nabla_{\theta} L(\theta)$$

  - ϵ → Learning rate
  - ∇_{θ} → Gradient of model parameters w.r.t. loss function
- Loss contours typically non-convex
  - Local minima and saddle points must be avoided

Multilayer Perceptron (MLP)

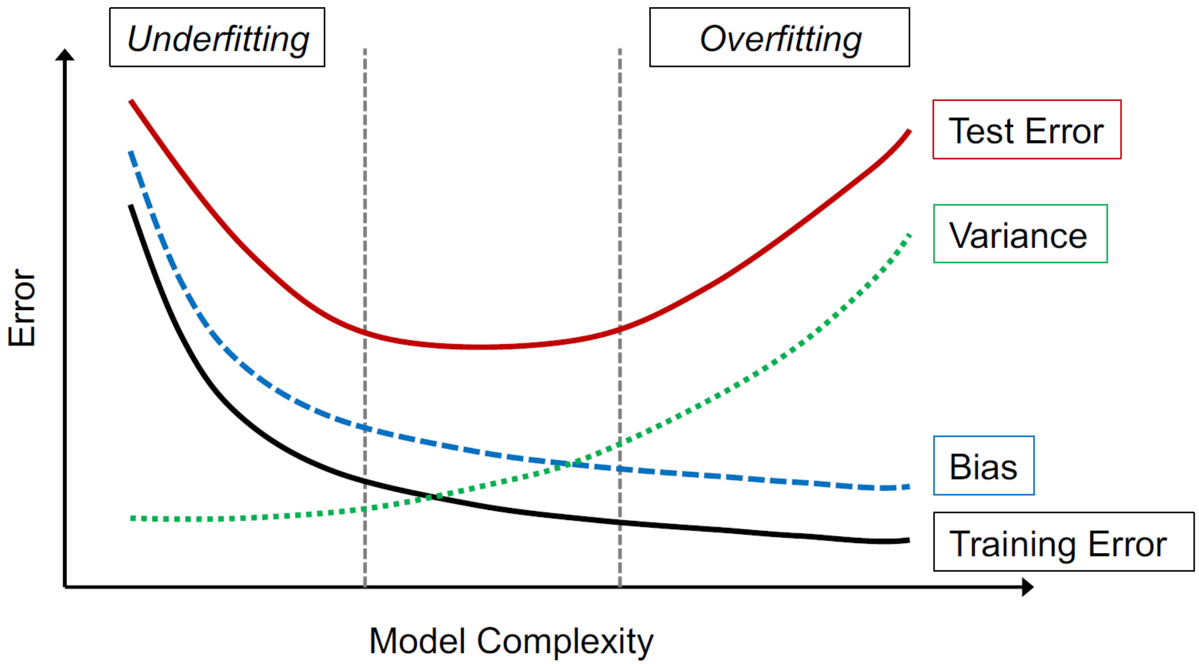
Model Training

- Backpropagation algorithm
  - Compute gradients of network weights w.r.t. loss function from last to first layer using the chain rule of derivatives
- Epoch
  - During one training epochs, the entire training set has been processed as **mini-batches**
- Loss functions
  - **Regression** → Mean squared error (MSE)
  - **Multilabel classification** → Binary Crossentropy
  - **Multiclass classification** → Categorical Crossentropy

Multilayer Perceptron (MLP)

Model Training

- Bias → Proportional to model error on training data
- Variance → Difference between training and test error
  - Measures how well model generalizes to new data



Own

Programming session

Fig-A2-13

References

Images

Fig-D2-1: Grollmisch S, Cano E. Improving Semi-Supervised Learning for Audio Classification with FixMatch. Electronics. 2021; 10(15):1807. https://doi.org/10.3390/electronics10151807, Fig. 2

Fig-D2-2: https://miro.medium.com/max/915/1*SJPacPhP4KDEB1AdhOFy_Q.png

Fig-D2-3: https://www.skampakis.com/wp-content/uploads/2018/03/simple_neural_network_vs_deep_learning.jpg

Fig-D2-4: https://pic4.zhimg.com/80/v2-057b248288a8af2f01272a956f862873_1440w.png