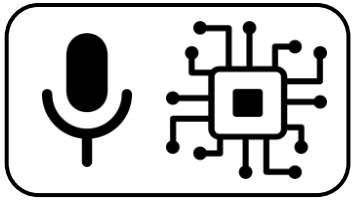
Computational Analysis of Sound and Music

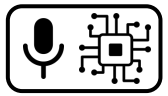


Deep Learning – Network Architectures

Dr.-Ing. Jakob Abeßer

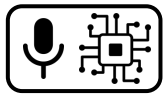
Fraunhofer IDMT

jakob.abesser@idmt.fraunhofer.de



Outline

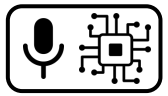
- **Convolutional Neural Networks (CNNs)**
  - Convolutional Layers
  - Resampling Layers
  - Batch Normalization Layers
  - Common Architectures
- Convolutional Recurrent Neural Networks (CRNN)
- Autoencoder / U-Net



Convolutional Neural Networks (CNNs)

Motivation

- Hierarchical feature learning of temporal-spectral patterns
    - Local temporal-spectral patterns → long-term patterns
- Shared weights
    - Fewer parameters
- Translation equivariance
    - Example: temporal shift of sound → same sound class



Convolutional Neural Networks (CNNs)

Convolutional Layers

- Components
  - Input feature $X \in \mathbb{R}^{M \times N}$
  - Kernel $K \in \mathbb{R}^{k \times k}$
  - Feature map S
- "Convolution" operation (actually a cross-correlation)

$$S_{i,j} = (K * X)_{i,j} = \sum_{u=1}^{k} \sum_{v=1}^{k} K_{u,v} \cdot X_{i+u,j+v}$$

- Parameters
  - Number of kernels
  - Kernel size

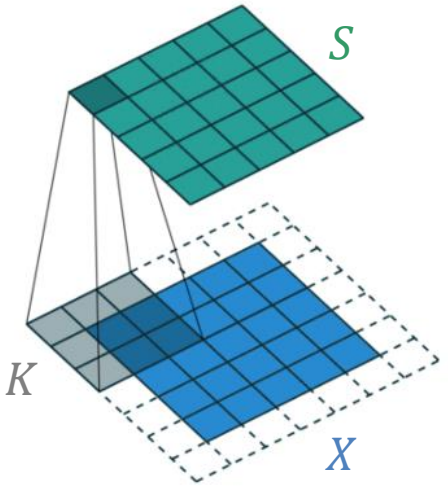


Fig-D3-1

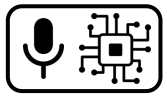
`tensorflow.keras.layers.Conv2D`

Convolutional Neural Networks (CNNs)

Convolutional Layers

- Stride s_k
  - Step size, with which K is shifted across X
- Optional symmetric zero-padding by p_k elements in X
  - „Valid“ convolution ($p_k = 0$) → S is smaller than X
  - „Same“ convolution ($p_k = \frac{k-1}{2}$) → S has the same shape as X
- Dilation rate d_k
  - Determines whether K
    - covers adjacent spatial elements in X ($d_k = 1$) or
    - skips one or more elements ($d_k > 1$)

`tensorflow.keras.layers.Conv2D`

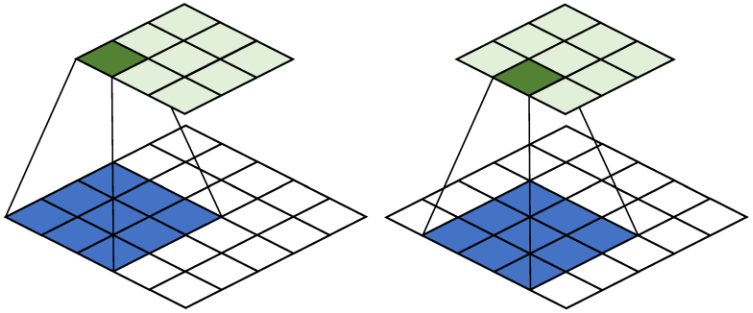


Convolutional Neural Networks (CNNs)

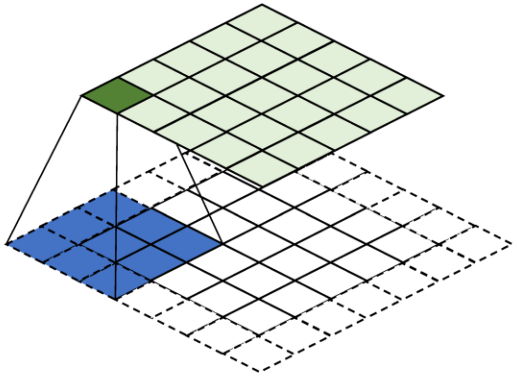
Convolutional Layers

- Examples

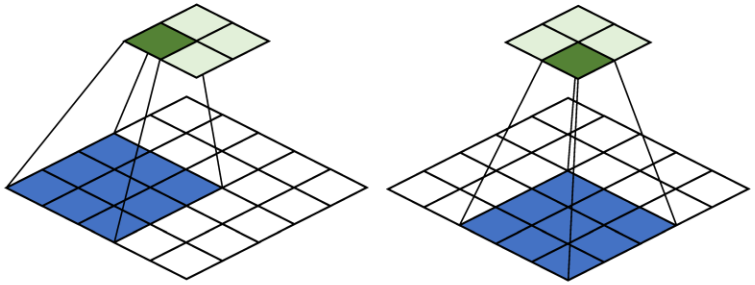
(1) Valid Convolution ($s_\mathrm{k} = 1, p_\mathrm{k} = 0, d_\mathrm{k} = 1$)



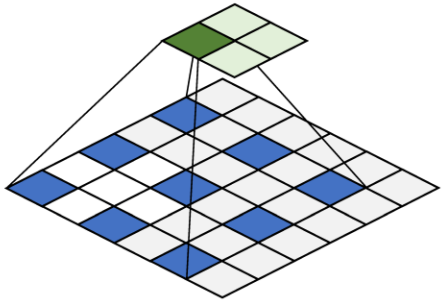
(2) Same Convolution ($s_\mathrm{k} = 1, p_\mathrm{k} = 1, d_\mathrm{k} = 1$)



(3) Valid Convolution ($s_\mathrm{k} = 2, p_\mathrm{k} = 0, d_\mathrm{k} = 1$)



(4) Valid Convolution ($s_\mathrm{k} = 1, p_\mathrm{k} = 0, d_\mathrm{k} = 2$)



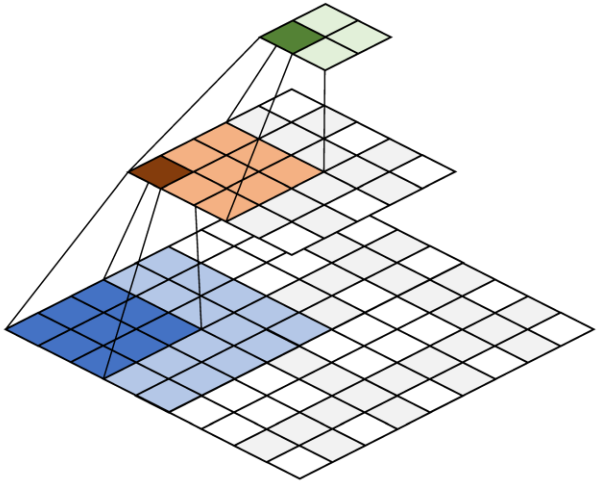
Own

Convolutional Neural Networks (CNNs)

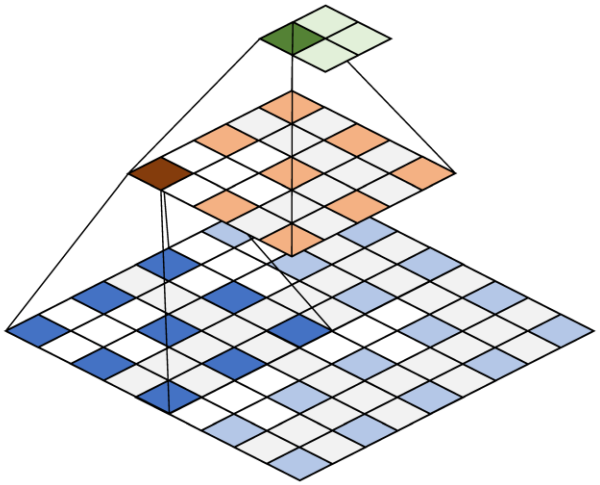
Resampling Layers

- Receptive field
  - Fraction of the input image, which a neuron in a layer can respond to.
  - Larger receptive field is generally beneficial for improving (but: risk overfitting)

(1) Non-Dilated Convolution ($d_k = 1$)



(2) Dilated Convolution ($d_k = 2$)

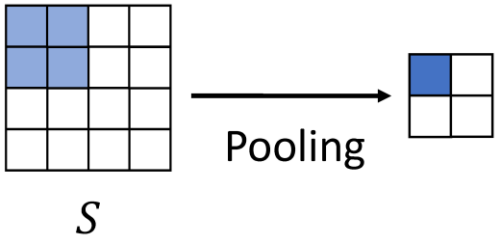


Own

Convolutional Neural Networks (CNNs)

Resampling Layers

- Pooling
  - Spatial downsampling of S
- Pooling operation
  - Max pooling (maximum)
  - Average pooling (mean)
- Increases translation invariance
- Reduces number of parameters



Own

`tensorflow.keras.layers.AveragePooling2D`

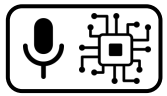
`tensorflow.keras.layers.MaxPooling2D`

Convolutional Neural Networks (CNNs)

Batch Normalization Layers

- Why?
  - Prevents internal data distribution shift (“covariate shift”)
  - Stabilizes and speeds up training
- Data Tensor $\mathcal{X} \in \mathbb{R}^{B \times F \times N}$
  - Mini-batch size B, number of frequency bins F, number of frames N
- Steps
  - Standardization (zero mean, unit variance) per mini-batch
  - Optional additional transformation with learnable scaling & shifting parameter

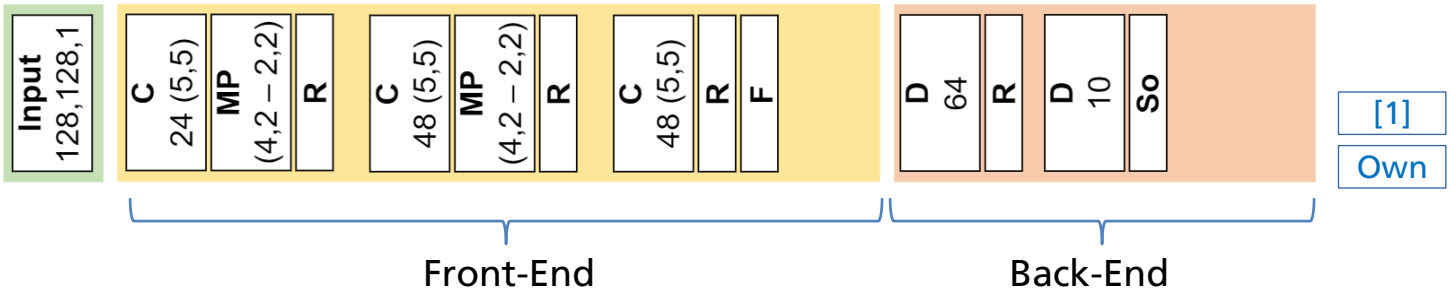
`tensorflow.keras.layers.BatchNormalization`



Convolutional Neural Networks (CNNs)

Common Architectures

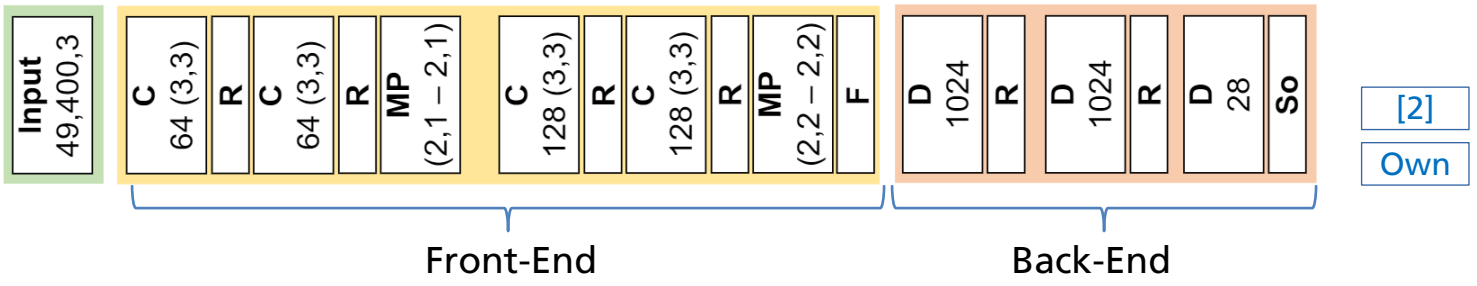
- Basic Architecture
  - Input layer (fixed patch size)
  - "Front-End": 3 Convolutional Blocks
    - Convolutional Layer (C) → Max Pooling (MP) → ReLU
  - Flatten
    - 4D tensor $\mathcal{X} \in \mathbb{R}^{B \times F \times N \times 1}$ → 2D tensor $\mathcal{X} \in \mathbb{R}^{B \times F \cdot N}$
  - "Back-End" → Dense + ReLU + Dense + Softmax



Convolutional Neural Networks (CNNs)

Common Architectures

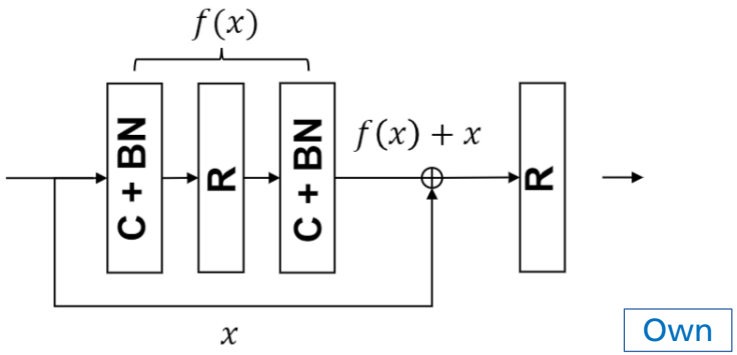
- "VGG" Architecture
  - Pairs of convolutional layers with small kernel size and intermediate activation function
    - Same receptive field as one CL with large kernel size but fewer parameters
    - Additional non-linearity makes model more expressive



Convolutional Neural Networks (CNNs)

Common Architectures

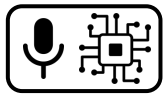
- Residual Networks
  - Deep neural networks suffer from vanishing gradient problem
  - Residual blocks with skip connection
    - Combine identity mapping with convolutional layers
  - Allow for steady gradient flow & deeper networks



Own

Outline

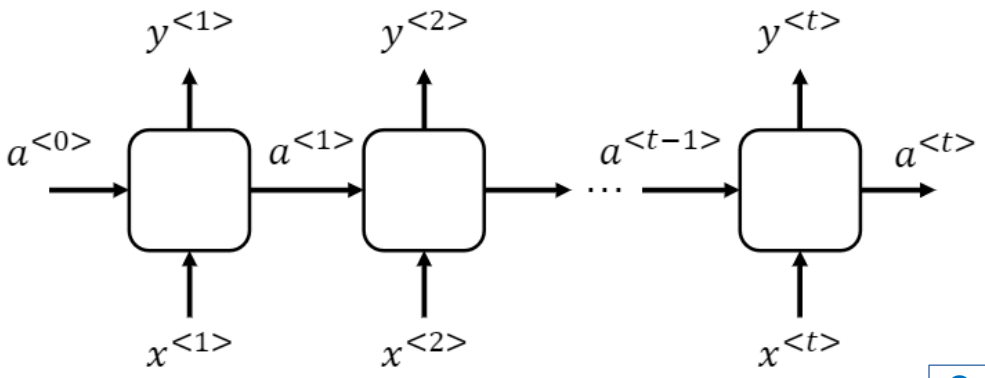
- Convolutional Neural Networks (CNNs)
    - Convolutional Layers
    - Resampling Layers
    - Batch Normalization Layers
    - Common Architectures
- **Convolutional Recurrent Neural Networks (CRNN)**
- Autoencoder / U-Net



Convolutional Recurrent Neural Networks (CRNNs)

Recurrent Neural Networks

- Commonly used to model sequential data e.g. in natural language processing (NLP)
- Recurrent layers
  - Additional time-dependent state variable $a^{<t>}$ (memory)
  - Weight sharing over time (CNN: over space)



Own

`tensorflow.keras.layers.SimpleRNN`

Convolutional Recurrent Neural Networks (CRNNs)

Recurrent Neural Networks

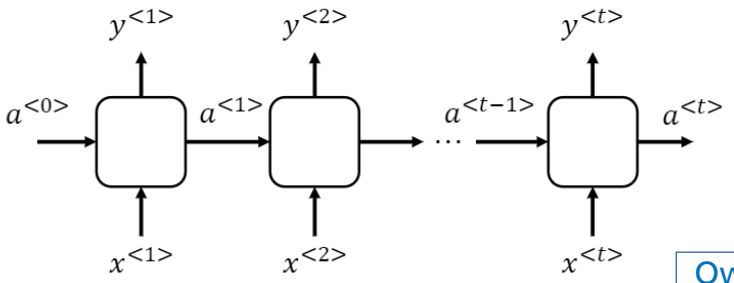
- Steps
  - State update

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}a^{<t>} + b_a)$$

  - Output update

$$\hat{y}^{<t>} = g_2\left(W_{ya}a^{<t>} + b_y\right)$$

  - Activation functions (g_1, g_1)
  - Trainable parameters (W_{aa}, W_{ax}, W_{ya}, b_a, b_y)



Own

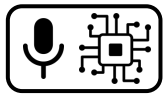
Convolutional Recurrent Neural Networks (CRNNs)

Recurrent Neural Networks

- Gated Recurrent Units (GRU)
    - Update gate to control information flow from previous state
    - Forget gate to control forgetting of past information
- Long Short-Term Memory (LSTM)
    - Input / Output / Forget gate
    - Increased computational complexity

`tensorflow.keras.layers.GRU`

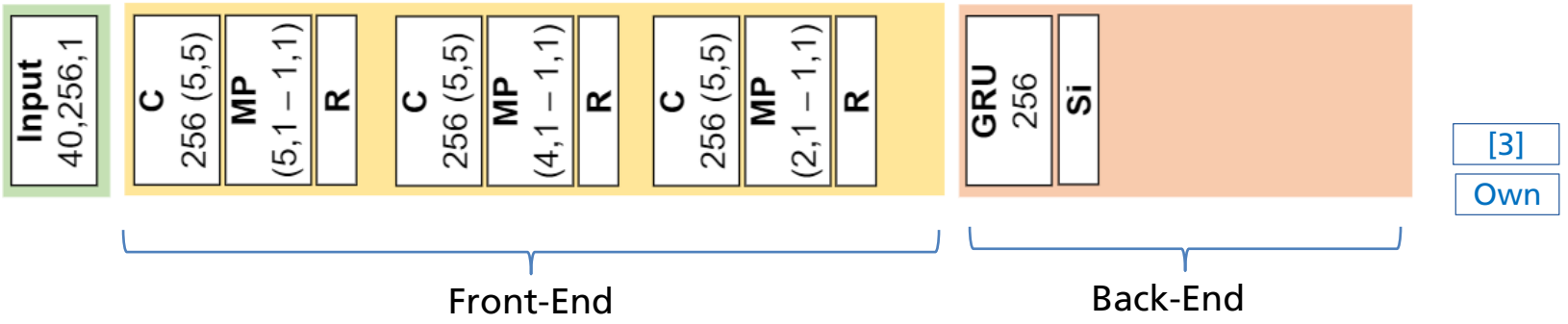
`tensorflow.keras.layers.LSTM`



Convolutional Recurrent Neural Networks (CRNNs)

Convolutional Recurrent Neural Networks

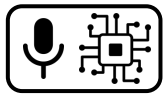
- Integrating recurrent layer in CNN backend
  - Front-end learns relevant local temporal-spectral patterns
  - Recurrent layers allow for temporal modeling thereof



[3]
Own

Outline

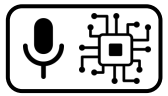
- Convolutional Neural Networks (CNNs)
    - Convolutional Layers
    - Resampling Layers
    - Batch Normalization Layers
    - Common Architectures
- Convolutional Recurrent Neural Networks (CRNN)
- **Autoencoder / U-Net**



Autoencoder / U-Net

Autoencoder

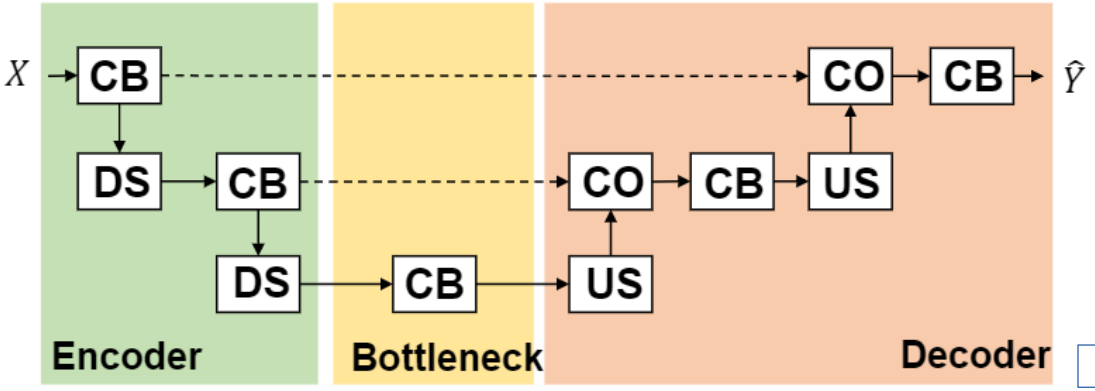
- Two-part structure
    - Encoder → Map input to lower-dimensional (latent) space
    - Decoder → Reconstruct input from latent space representation
- Need to learn compressed/efficient feature representation (encodings)
- Applications
    - Compression
    - Denoising (map noisy input to clean input)
    - Anomaly detection (high reconstruction error indicates anomaly)
    - Generative models (using decoder)
- Main advantage → unsupervised training possible



Autoencoder / U-Net

U-Net

- Components
  - Convolutional blocks (CB)
  - Down-sampling (DS) in encoder → Up-sampling (US) in decoder
  - Additional skip-connections
- Applications
  - Music transcription & source separation



Own

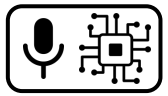
Programming session

Fig-A2-13

References

Images

Fig-D3-1: https://www.wandb.com/articles/intro-to-cnns-with-wandb



References

References

[1] J. Salamon and J. P. Bello, “Deep convolutional neural networks and data augmentation for environmental sound classification,” IEEE Signal Processing Letters, vol. 24, no. 3, pp. 279–283, 2017

[2] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, “Deep convolutional neural networks and data augmentation for acoustic event recognition,” in Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH), San Francisco, CA, USA, 2016, pp. 2982–2986.

[3] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 25, no. 6, pp. 1291–1303, 2017.