TUBAF

The University of Resources.
Since 1765.

Scientific Computing Project

Winter Semester 2024/2025

Lars Wunderlich, Parsa Besharat, Toni Sand

TU Bergakademie Freiberg

May 19, 2025

Content

Task

1. Task

- implement a **Convolutional Neural Network**

1. Task

- implement a **Convolutional Neural Network**
- classify images from 10 categories

1. Task

- implement a **Convolutional Neural Network**
- classify images from 10 categories
- network architecture: **VGG-16**

1. Task

- implement a **Convolutional Neural Network**
- classify images from 10 categories
- network architecture: **VGG-16**
- hyperparameter tuning via **k-fold cross-validation**

1. Task

- implement a **Convolutional Neural Network**
- classify images from 10 categories
- network architecture: **VGG-16**
- hyperparameter tuning via **k-fold cross-validation**
- **evaluation** with performance measurements

Description of the Dataset

2. Dataset

10 classes of images:

2. Dataset

10 classes of images:

1. bottles	2. mugs/cups	3. spoons	4. knifes	5. forks
6. shoes	7. t-shirts	8. plants	9. chairs	10. bikes

2. Dataset

10 classes of images:

1. bottles 2. mugs/cups 3. spoons 4. knifes 5. forks
6. shoes 7. t-shirts 8. plants 9. chairs 10. bikes

(a) 10. bikes

(b) 2. mugs/cups

(c) 3. spoons

2. Dataset

Augmentation by:

2. Dataset

Augmentation by:

- horizontal flip
- vertical flip
- crop
- rotation
- blur
- change of brightness
- change of contrast
- change of color

2. Dataset

Augmentation by:

- horizontal flip
- vertical flip
- crop
- rotation
- blur
- change of brightness
- change of contrast
- change of color

- application of up to 4 different augmentations per image

2. Dataset

Augmentation by:

- horizontal flip
- vertical flip
- crop
- rotation
- blur
- change of brightness
- change of contrast
- change of color

- application of up to 4 different augmentations per image
- random intensity

2. Dataset

Augmentation by:

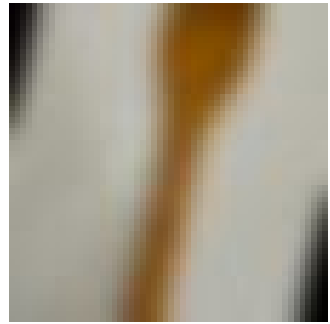
- horizontal flip
- vertical flip
- crop
- rotation
- blur
- change of brightness
- change of contrast
- change of color

- application of up to 4 different augmentations per image
- random intensity
- normalization to 56x56 pixels

2. Dataset

(a) 10. bike

(b) 1. bottles



(c) 3. spoons

Figure: Normalized images of the augmented database with different transformations

Architecture of the Network

3. Architecture

- network architecture: **VGG-16**:

3. Architecture

- network architecture: **VGG-16**:
  - deep but uniform architecture

3. Architecture

- network architecture: **VGG-16**:
  - deep but uniform architecture
  - many convolutional layers with kernel size 3 followed by a pooling layer

3. Architecture

- network architecture: **VGG-16**:
  - deep but uniform architecture
  - many convolutional layers with kernel size 3 followed by a pooling layer
  - simplified: just 4 repetitions of these layers with max-pooling

3. Architecture

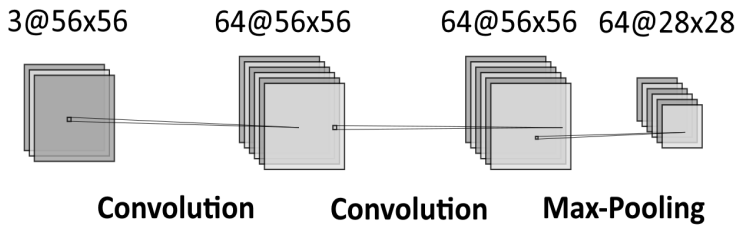


Figure: Inital sequence of 2 convolutional layers with max-pooling

3. Architecture

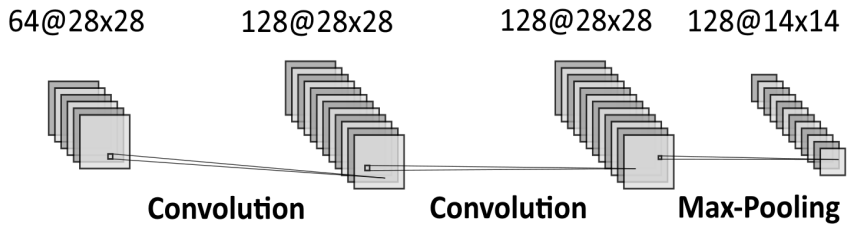


Figure: First repetition with 2 convolutional layers and max-pooling

3. Architecture

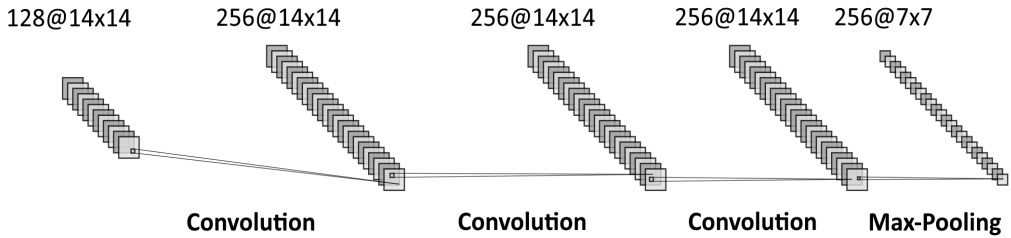


Figure: Second repetition with 3 convolutional layers and max-pooling

3. Architecture

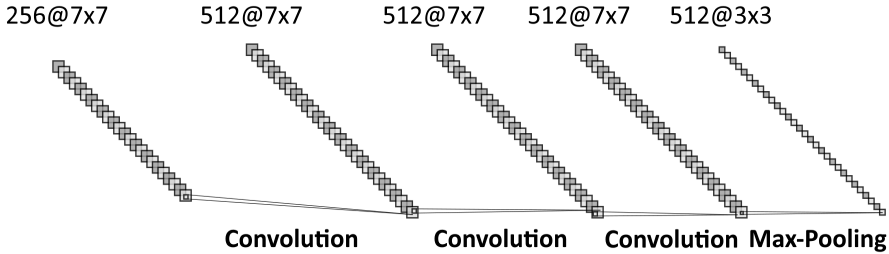


Figure: Third repetition with 3 convolutional layers and max-pooling

3. Architecture

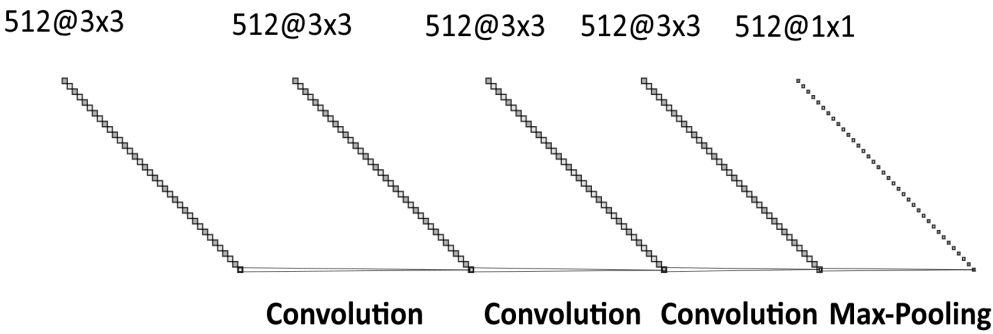


Figure: Fourth repetition with 3 convolutional layers and max-pooling

3. Architecture

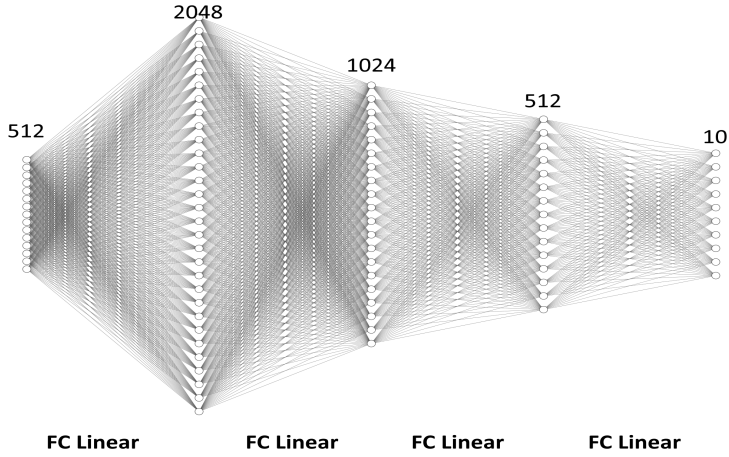


Figure: Fully connected linear layers at the end

Training

4. Training

- hyperparameter tuning via **3-fold cross-validation**

4. Training

- hyperparameter tuning via **3-fold cross-validation**
- used for finding best learning rate, batch size for **ADAM** and **SGD**

4. Training - ADAM

- ADAM = Adaptive Moment Estimation

[1] source: A Method for Stochastic Optimization - https://arxiv.org/abs/1412.6980

4. Training - ADAM

- ADAM = Adaptive Moment Estimation
- adapts learning rate for each parameter

[1] source: A Method for Stochastic Optimization - https://arxiv.org/abs/1412.6980

4. Training - ADAM

- ADAM = Adaptive Moment Estimation
- adapts learning rate for each parameter
- uses estimates of first and second moments of the gradient (mean, uncentered variance)

[1] source: A Method for Stochastic Optimization - https://arxiv.org/abs/1412.6980

4. Training - ADAM

- ADAM = Adaptive Moment Estimation
- adapts learning rate for each parameter
- uses estimates of first and second moments of the gradient (mean, uncentered variance)
- leads to faster convergence, better on complex and noisy datasets [1]

[1] source: A Method for Stochastic Optimization - https://arxiv.org/abs/1412.6980

4. Training - ADAM

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta L(\theta_t)$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_\theta L(\theta_t))^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

m_t : first moment (mean of gradients)
v_t : second moment (uncentered variance of gradients)
β_1, β_2 : decay rates for moments
$\hat{m}_t, \hat{v}_t$: bias-corrected estimates
ϵ : small constant (for numerical stability)

4. Training - SGD

- SGD = Stochastic Gradient Descent

[2] source: Mathematics of Machine Learning lecture script - B. Sprungk

4. Training - SGD

- SGD = Stochastic Gradient Descent
- updates model parameters using current gradient

[2] source: Mathematics of Machine Learning lecture script - B. Sprungk

4. Training - SGD

- SGD = Stochastic Gradient Descent
- updates model parameters using current gradient
- simple and efficient

[2] source: Mathematics of Machine Learning lecture script - B. Sprungk

4. Training - SGD

- SGD = Stochastic Gradient Descent
- updates model parameters using current gradient
- simple and efficient
- but sensitive to learning rate and slower [2]

[2] source: Mathematics of Machine Learning lecture script - B. Sprungk

4. Training - SGD

Stochastic Gradient Descent [3] uses one randomly selected gradient per step:

[3] source: Mathematics of Machine Learning lecture script - B. Sprungk

4. Training - SGD

Stochastic Gradient Descent [3] uses one randomly selected gradient per step: Given a starting vector $\mathbf{w}_0 \in \mathbb{R}^p$ and an objective function

$$F(w) = \frac{1}{m} \sum_{i=1}^{m} f_i(w), \qquad f_i : \mathbb{R}^p \to \mathbb{R},$$

calculate for $k = 0, 1, 2, \ldots$ the iterates $\mathbf{w}_{k+1}$ as follows:

[3] source: Mathematics of Machine Learning lecture script - B. Sprungk

4. Training - SGD

Stochastic Gradient Descent [3] uses one randomly selected gradient per step:
Given a starting vector $\mathbf{w}_0 \in \mathbb{R}^p$ and an objective function

$$F(w) = \frac{1}{m} \sum_{i=1}^{m} f_i(w), \qquad f_i : \mathbb{R}^p \to \mathbb{R},$$

calculate for $k = 0, 1, 2, \ldots$ the iterates $\mathbf{w}_{k+1}$ as follows:
1. draw realization $i_k \in [m]$ of the uniformly random index variable

$$I_k \sim U([m]), \qquad [m] := \{1, \ldots, m\}$$

where the I_k, $k \in \mathbb{N}$, are stochastically independent,

[3] source: Mathematics of Machine Learning lecture script - B. Sprungk

4. Training - SGD

Stochastic Gradient Descent [3] uses one randomly selected gradient per step:
Given a starting vector $\mathbf{w}_0 \in \mathbb{R}^p$ and an objective function

$$F(w) = \frac{1}{m}\sum_{i=1}^{m} f_i(w), \qquad f_i : \mathbb{R}^p \to \mathbb{R},$$

calculate for $k = 0, 1, 2, \ldots$ the iterates $\mathbf{w}_{k+1}$ as follows:
1. draw realization $i_k \in [m]$ of the uniformly random index variable

$$I_k \sim U([m]), \qquad [m] := \{1, \ldots, m\}$$

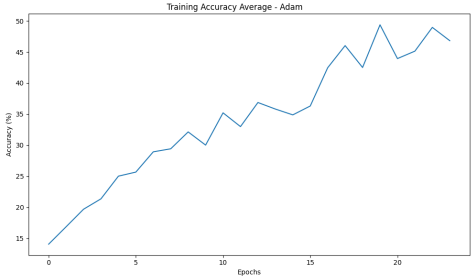
where the I_k, $k \in \mathbb{N}$, are stochastically independent,
2. for a given deterministic step size $\eta_k > 0$, calculate

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \nabla f_{i_k}(\mathbf{w}_k).$$

[3] source: Mathematics of Machine Learning lecture script - B. Sprungk

Results

5. Results



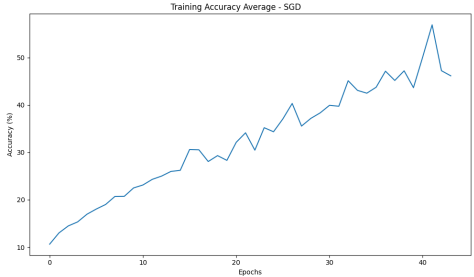


Figure: Average accuracy of ADAM and SGD with respect to training epochs

5. Results

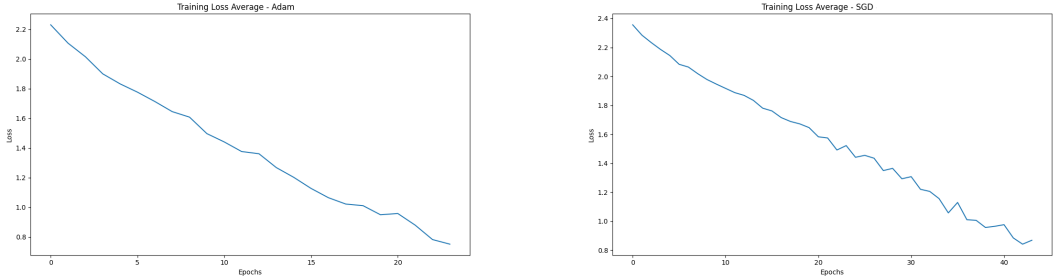


Figure: Average loss of ADAM and SGD with respect to training epochs

5. Results

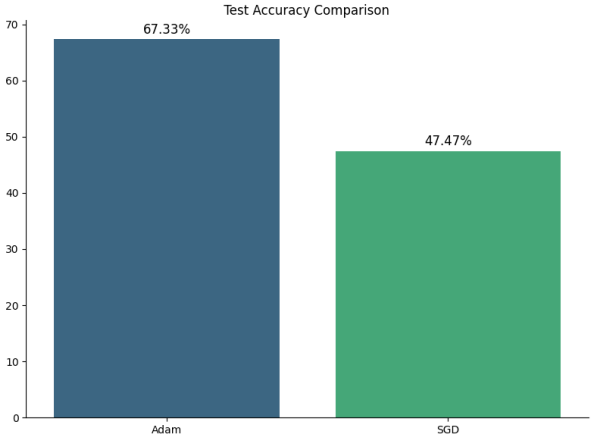


Figure: Comparison of the accuracy of both optimizers over test data

5. Result

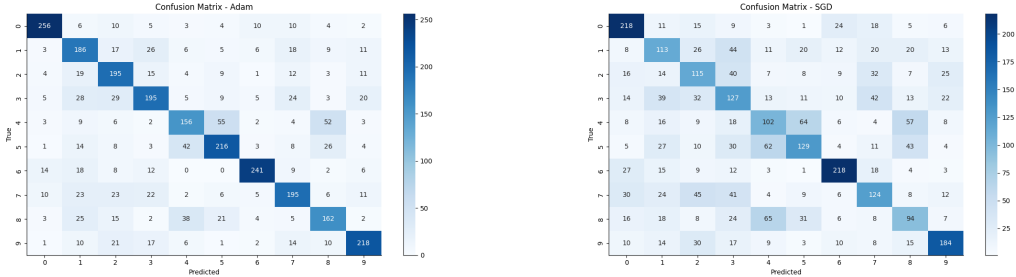


Figure: Confusion matrices for ADAM and SGD for the test data

5. Result

- possible ways for improvement:

5. Result

- possible ways for improvement:
  - increase image resolution

5. Result

- possible ways for improvement:
  - increase image resolution
  - having more computational power / time

5. Result

- possible ways for improvement:
  - increase image resolution
  - having more computational power / time
  - using more repetitions in the architecture of the VGG-16