Neural Networks 3 - Neural Networks

18NES2 - Lecture 4, Winter semester 2025/26

Zuzana Petříčková

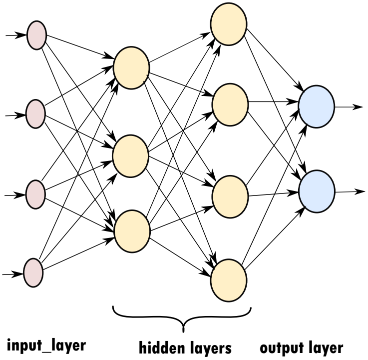
October 14, 2025

Neural Networks 2 - Practical Tasks and Examples

What We Covered Last Time

Multi-Layer Neural Network Model (Multi-Layer Perceptron, MLP)

- Training process
- Python Libraries for Machine Learning and Deep Learning - practical examples
- Training workflow in Keras: a complete example (partly as homework)
- Homework assignment



What We Covered Last Time

Training workflow in Keras: a complete example

- We can quickly revisit this example today — we didn't have time to finish it last time.

Homework assignment

- The assignment was due **today**.
- Those who have submitted can arrange a short consultation with me this week.
- **Points will be awarded only to those who complete the consultation by the end of this week.**

Practical Examples of MLP on Different Task Types

We will look at three basic machine learning tasks:

- **Binary classification** – IMDB movie reviews
  - Text data $\rightarrow$ sentiment analysis (positive / negative)
  - Representation: Bag-of-Words (vector of word occurrences)
- **Multiclass classification** – Fashion-MNIST
  - Image data $\rightarrow$ clothing type recognition
  - Representation: pixel intensity matrix (grayscale images)
- **Regression** – Boston Housing dataset
  - Tabular numerical data $\rightarrow$ prediction of house prices
  - Focus on model evaluation (cross-validation)

Together, these examples cover the main data types used in deep learning: **text**, **image**, and **numerical** data.

- What about **time series**? $\rightarrow$ for a MLP example, see materials for 18NES1, we will return to this type of data later.

Neural Networks 2 - Practical Tasks and Examples

Example: Binary Classification — IMDB Dataset

binary_classification_imdb.ipynb

- Commented example (full training workflow in Keras + additional practice tasks)

About the IMDB Dataset

- **50,000** movie reviews: **25,000 train** / **25,000 test**
- Reviews represented as **sequences of integer word indices** (by frequency)
- Use only the most frequent words (e.g., **top 10,000**) to limit vocabulary size
- **Target:** sentiment (0 = negative, 1 = positive)

Model choice

- For text classification, an **MLP** (on BoW) can serve as a **solid baseline**
- Note: More specialized models (Embedding + **1D-CNN**/**RNN**/**Transformer**) often perform better on text.

Now we will implement a minimal MLP as the baseline model.

Integer Word Encoding (IMDB Dataset Example)

Original reviews (toy example)

Review 1: *"The movie was great"*
Review 2: *"The movie was not good"*

Vocabulary (top 6 words)

[the, movie, was, great, not, good] $\Rightarrow$ [1, 2, 3, 4, 5, 6]

Integer-encoded representation

	Word indices
Review 1	[1, 2, 3, 4]
Review 2	[1, 2, 3, 5, 6]

(Each review becomes a sequence of integers; lengths differ before padding.)

Example: Binary Classification — IMDB Dataset

- Movie reviews are represented as sequences of word indices
  - Indices assigned to words according to word frequency in the corpus
  - Only the most frequent words are used
- **Problem:** Reviews differ in length — we need a fixed-size input representation to use multi-layer (dense) neural networks
- **Possible solutions:**
  - Padding / truncation of sequences to the same length
  - Bag-of-Words (BoW) or TF–IDF representation

Bag-of-Words Representation (BoW)

Original reviews (toy example)

Review 1: *"The movie was great"*
Review 2: *"The movie was not good"*

Vocabulary (top 6 words)

[the, movie, was, great, not, good] $\Rightarrow$ [0, 1, 2, 3, 4, 5]

Bag-of-Words representation (binary / multi-hot encoding)

	the	movie	was	great	not	good
Review 1	1	1	1	1	0	0
Review 2	1	1	1	0	1	1

(Each review is represented by a fixed-length binary vector of vocabulary size.)

Bag-of-Words Representation (BoW)

- Simplest way to obtain a fixed-length input vectors
- Each review is converted into a vector of word counts (or binary values)
- The vector length equals the vocabulary size (e.g., 10,000)

Advantages

- Easy to implement and interpret
- Works surprisingly well for simple tasks (e.g., sentiment or topic classification)

Bag-of-Words Representation (BoW)

Limitations

- Does not take into account the order or position of words in the text
- Loses information about context and word relationships
- Produces large, sparse matrices with high memory requirements
- **No notion of similarity:** In this space, all words are equally distant — the model cannot capture that *"good"* and *"excellent"* are semantically related

Better text representations?

- Techniques such as **TF–IDF**, **word embeddings** (Word2Vec, GloVe) or **contextual embeddings** (BERT) address these issues

(We will discuss these more advanced representations in detail later in the semester.)

Example: Binary Classification — IMDB Dataset

Model setup

- Start with a relatively small model and use a larger batch size
- Sigmoid activation in the output layer
- ReLU (or *tanh*) activations in hidden layers
- Loss function: `BinaryCrossentropy`, metric: `BinaryAccuracy`

Observations

- Test accuracy around 85%
- The model tends to overfit quickly (validation loss increases) $\rightarrow$ Try reducing the number of epochs, using early stopping, or applying regularization techniques

Example: Binary Classification — IMDB Dataset

Summary

1. For binary classification, use the `BinaryCrossentropy` loss (MSE can also be used) and the `BinaryAccuracy` metric. The output layer uses a sigmoid activation.
2. Each review is represented by a fixed-length Bag-of-Words vector (binary or count-based representation of the most frequent words).
3. Bag-of-Words is a simple and effective way to represent text for dense neural networks. (Alternatives: TF–IDF, word embeddings, etc.)
4. Validation data help assess how well the model learns and generalizes.
5. The IMDB dataset is ideal for illustrating binary text classification and serves as a bridge to more advanced natural language processing (NLP) models.

Neural Networks 2 - Practical Tasks and Examples

Example: Multiclass Classification Task – Fashion MNIST

binary_classification_fashion_mnist.ipynb

- Commented example (full training workflow in Keras + additional practice tasks)

About the MNIST Dataset

- **70,000** grayscale images of clothing items from **10 categories** (e.g., T-shirt, trousers, bag), size **28×28**
- Centered objects of similar size
- **60,000** training images / **10,000** test images
- **Target:** label **0–9** (10 classes)

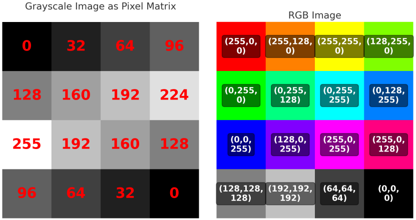
Model choice

- For image classification, a **MLP** is a clear **baseline**.
- Note: **CNNs** usually perform better on images by exploiting spatial structure (convolutions/pooling).

Now we will implement a minimal MLP (and later compare with a CNN).

Digital Image Representation

- A digital image is a matrix (tensor) of numerical values (pixel intensities)
- Each pixel (short for "picture element") describes the color at a specific position in the image.



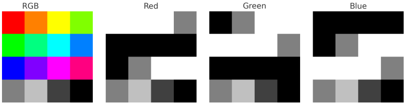
Grayscale Image

- Each pixel is a single value indicating brightness (e.g., $0 =$ black, $255 =$ white).
- For machine learning, pixel values are usually normalized to the interval $[0, 1]$.

Digital Image Representation

Color Image (RGB)

- Each pixel consists of three components: R (red), G (green), B (blue).
- The image is represented as a 3D tensor of shape (height $\times$ width $\times$ 3).
- These components are called color channels.



Preprocessing of image data:

- Resizing images to a fixed size
- Normalizing pixel values (e.g., to $[0, 1]$ or $[-1, 1]$)
- For simple models (e.g., MLP), images must be **vectorized** — converted to 1D vectors

Example: Multiclass Classification Task – Fashion MNIST

- 70,000 grayscale images of of clothing items from 10 categories (e.g., shirts, shoes, bags) (28x28)
- Centered objects, all of similar size
- 60,000 training images, 10,000 test images
- Output label: 0–9 (10 classes)
- Data characteristics:
  - All images have the same size $\rightarrow$ no need to standardize shape
  - Input data are 3D $\rightarrow$ must be vectorized
  - Pixel values range from 0...255 $\rightarrow$ must be normalized to $[0, 1]$ or $[-1, 1]$

Example: Multiclass Classification Task – Fashion MNIST

Model setup

- Softmax activation function in the output layer
- ReLU (or tanh) activation in hidden layers
- Loss: **SparseCategoricalCrossentropy**, Metric: **SparseCategoricalAccuracy** (if labels are integers)
- Loss: **CategoricalCrossentropy**, Metric: **CategoricalAccuracy** (if labels are one-hot vectors)

Typical observations

- Test accuracy is typically around 79-80% for a simple MLP
- Training and validation accuracy close $\Rightarrow$ model **generalizes well** (no overfitting)
- Both accuracies relatively low $\Rightarrow$ possible **underfitting**
- Accuracy may improve with a larger or deeper model,more epochs or better learning rate, or using a **CNN** instead of MLP

Example: Multiclass Classification – Fashion MNIST

Summary

1. For multiclass classification, use the **CategoricalCrossentropy** or **SparseCategoricalCrossentropy** loss, and the **Accuracy** metric.
2. The output layer uses the softmax activation function.
3. Image data should be normalized (pixel values scaled to $[0, 1]$ or $[-1, 1]$).
4. For MLP models, input images must be flattened to 1D vectors.
5. The Fashion MNIST dataset is ideal for illustrating multiclass classification and serves as a bridge to convolutional neural networks (CNNs).

Neural Networks 2 - Practical Tasks and Examples

Example: Regression Task — Boston Housing Data

regression_boston_housing.ipynb

- Commented example (full training workflow in Keras + additional practice tasks)

About the Boston Housing Dataset

- Classic machine learning benchmark dataset for regression
- **506** samples, **13** numerical features (e.g., RM, LSTAT, PTRATIO, NOX, CRIM)
- Target: median house value (**MEDV**) in $1,000s
- 404 training examples, 102 testing
- Features are on very different scales

Model choice

- For small tabular regression tasks, an **MLP is a strong baseline** (DL/NN context).
- In classical ML, **tree-based ensembles** (e.g., Gradient Boosting) are often competitive or superior.

Example: Regression Task — Boston Housing Data

Feature meanings

- `CRIM` — per capita crime rate by town
- `ZN` — proportion of residential land zoned for lots > 25,000 sq.ft.
- `INDUS` — proportion of non-retail business acres per town
- `CHAS` — Charles River dummy (1 if tract bounds river; else 0)
- `NOX` — nitric oxides concentration (ppm)
- `RM` — average number of rooms per dwelling
- `AGE` — proportion of owner-occupied units built prior to 1940
- `DIS` — weighted distances to Boston employment centres
- `RAD` — index of accessibility to radial highways
- `TAX` — full-value property-tax rate per $10,000
- `PTRATIO` — pupil–teacher ratio by town
- `B` — $1000 \times (B_k - 0.63)^2$, where B_k is proportion of Black residents (historically sensitive)
- `LSTAT` — % lower status of the population

Example: Regression Task — Boston Housing Data

Data characteristics:

- Features have very different ranges of values →
  - Recommended preprocessing: **normalization** of features
  - **StandardScaler** - normalize each feature (zero mean, unit variance)
- The dataset is small (404 training samples) →
  1. The model must be small to avoid overfitting
  2. A single test set is not sufficient for reliable evaluation → use **cross-validation**

Cross-Validation

- Allows us to estimate how well the model generalizes even when the dataset is relatively small
- A generalization of the train/test split approach
- Useful for reliable model and hyperparameter comparisons

Monte Carlo Cross-Validation: random repeated splitting

1. For $i = 1, \ldots, k$:
   - Randomly split dataset T into T_1 (training) and T_2 (test), e.g. 70:30
   - Train the model on T_1, evaluate on T_2
   - Record the test error
2. Compute mean and standard deviation of errors over k runs (typically $k = 100$)

Cross-Validation

K-Fold Cross-Validation

- Compared to Monte Carlo, it systematically covers the entire dataset (no sample is left out)

Basic principle:

1. Split training data T into k equally sized disjoint subsets $T_1, \ldots, T_k$
2. For $i = 1, \ldots, k$:
    - Train on $T \setminus T_i$, evaluate on T_i
    - Record the test error
3. Compute the average and standard deviation over all k runs (commonly $k = 10$)

Example: Regression Task — Boston Housing Data

Model setup

- For small datasets, 1–2 hidden layers are sufficient
- Linear activation function in the output layer
- ReLU (or *tanh*) activations in hidden layers
- Loss function: **MeanSquaredError (MSE)** Metrics: **MAE**, **MSE** (https://keras.io/api/metrics/regression_metrics)

Observations

- If the model is too large or trained too long, it will overfit (validation/test error increases)

Example: Regression Task — Boston Housing Data

Summary

1. For regression tasks, use the **MSE** loss and regression metrics (**MSE**, **MAE**, ...). The output layer uses a linear activation.
2. If input features have different value ranges, normalize each feature.
3. The smaller the training dataset, the smaller the model should be to avoid overfitting.
4. If training continues for too long, overfitting occurs — validation error increases.
5. When data are limited, k-fold cross-validation gives a better estimate of model performance.

- The Boston Housing dataset is a classic regression benchmark and serves as a reminder that deep learning methods can also be applied to traditional machine learning tasks.

Summary: When to Use MLP Models

Key points:

- Multilayer Perceptrons (MLPs) work well for **smaller datasets** with **numerical features**.
- They are suitable for tasks where the input can be represented as a **fixed-size feature vector**.
- MLPs serve as a strong **baseline model** for more complex data types such as:
  - **Images** (before using CNNs),
  - **Text** (before using RNNs or Transformers),
  - **Time series** (before using sequence models).
- Provide a simple and interpretable starting point for comparison with more advanced architectures.

→ *MLPs are an essential building block in deep learning — simple, fast, and a reliable baseline.*

2nd Graded Homework: From IMDB (Binary) to Reuters (Multiclass)

Goal: Start from your **IMDB (binary)** notebook and refactor it to **Reuters (multiclass)**. Use the **Fashion-MNIST** setup as inspiration for the multiclass head (softmax, one output per class).

Dataset:

- **keras.datasets.reuters** — newswire topic classification (multiclass).
- Use the **same data representation** as in IMDB (Bag-of-Words); limit vocabulary (e.g., `num_words=10000`).

Modeling hints:

- Replace the binary output (**sigmoid**) with **softmax** over C classes.
- Loss/metrics: **SparseCategoricalCrossentropy** + **SparseCategoricalAccuracy** (for integer labels).

2nd Graded Homework: From IMDB (Binary) to Reuters (Multiclass)

Requirements

- Use an analogous preprocessing and representation as in IMDB (BoW).
- Keep the pipeline: load $\rightarrow$ preprocess $\rightarrow$ build $\rightarrow$ train $\rightarrow$ evaluate.
- Show training curves (loss/accuracy) and discuss overfitting/underfitting.
- Report **accuracy** and include a **confusion matrix**.
- Try **3+** hyperparameter changes (e.g., vocabulary size, layers/units, optimizer/learning rate, early stopping).
- Provide a short summary of your findings in the notebook.
- Show a few misclassified texts and briefly comment on them.
- Compare the results with IMDB / Fashion-MNIST.

2nd Graded Homework: From IMDB (Binary) to Reuters (Multiclass)

Submission

- Submit the notebook by **Oct 21, 2025**.
- **Consultation required by Oct 24, 2025** to receive points (short discussion after lab or individually).