University of Essex Online

Machine Learning

Assessment from Unit 11

Agenda

1. General introduction to the task and dataset
2. Validation set
3. Convolution Neural Network
4. Architecture of the model
5. Model details: activation function
6. Model details: loss function
7. Strategy of the model's design
8. Conclusion and outcomes
9. Reflections and lessons learnt
10. References
11. Appendix 1. Confusion matrix
12. Appendix 2. Code listing

General introduction to the task and dataset

General introduction

- The task is to create an artificial neural network recognizing pictures from ten different categories based on the CIFAR-10 dataset.
- CIFAR-10: 60,000 RGB pictures of 32px x 32px in ten given categories.

Explanatory Data Analysis

- The dataset is initially divided into two groups: a training set (50,000 elements) and a test set (10,000 elements).
- The data is structured in four arrays (or two tuples) – see Pic.1.
- The dataset is consistent and complete.

	Training	*Testing*
Picture	50,000 elements of 32x32x3	10,000 elements of 32x32x3
Label	50,000 elements of 1	10,000 elements of 1

Pic. 1. Dataset structure.

Validation set

The importance of maintaining a separate validation set

- Independent indicator of an accuracy gaining rate by the model.
- A trigger to finish the training procedure after a number of ineffective training epochs.
- Efficient method of overtraining detecting.

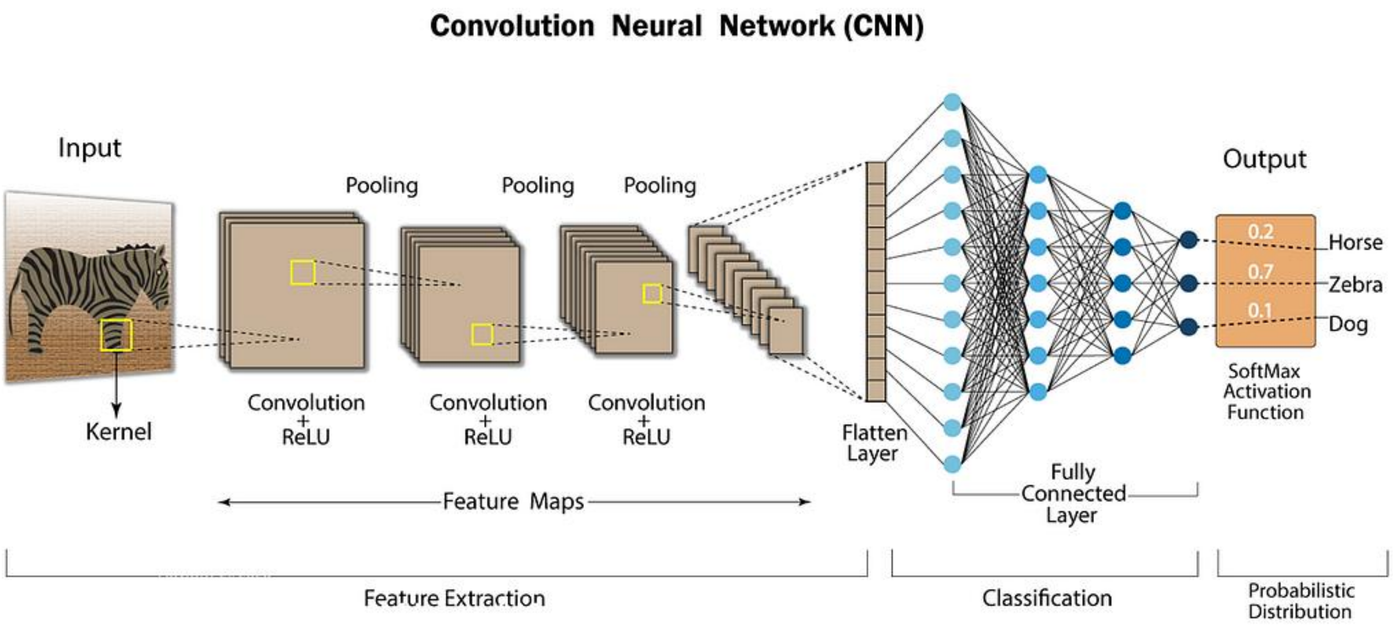
The proportion of training / validation / testing sets

- „To train multilayer perceptrons is more art than science” (M. Kubat, *An Introduction to Machine Learning*)
- As there is no general rule, the scholars seem to generally agree that the bigger the dataset, the share of a training part should tower above the rest.
- However, the proportion for this task will be 2/3 (66%) training, 1/6 (17%) validation, 1/6 (17%) testing.

	Training	*Validation*	*Testing*
Picture	40,000 elements of 32x32x3	10,000 elements of 32x32x3	10,000 elements of 32x32x3
Label	40,000 elements of 1	10,000 elements of 1	10,000 elements of 1

Convolutional Neural Network

- For the task given in the assignment, we will use Convolutional Neural Network—a type of neural network where a multi-layer perceptron is followed by a number of convolutional filters and spatial reduction.
- These layers preprocess the input image for subsequent deep networks: they detect important features, reduce noise and downscale where the amount of provided details is excessive.



Pic. 2. Convolutional Neural Network layers. Author: Pratham Modi
(https://medium.com/@prathammodi001/convolutional-neural-networks-for-dummies-a-step-by-step-cnn-tutorial-e68f464d608f)

Architecture of the model

- As cited in previous slide, tuning a neural network is more an art than a procedure. In this task I decided to try a few different architecture structures to examine obtained results.

	Layers	*Size*	
	Input layer	32x32x3	
baseline model	Convolutional 2D no.1	29x29x32	
	MaxPooling2D no.1	14x14x32	
	Convolutional 2D no.2	11x11x32	+ 1st added modification
	MaxPooling2D no.2	5x5x32	
baseline model	Flattening	As above	
	Hidden neural layer	256 neurons	
	Hidden neural layer	256 neurons	+ 2nd added modification
baseline model	Output neural layer	10 neurons	

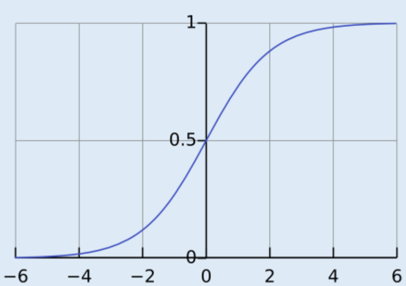
- Accuracy of the model with different architecture and fixed remaining hyperparameters:
  - Baseline model: accuracy 0.607 (6 epochs)
  - Baseline model w/ 1st modification: accuracy **0.659** (9 epochs)
  - Baseline model w/ 2nd modification: accuracy 0.662 (7 epochs)

Model details: activation function

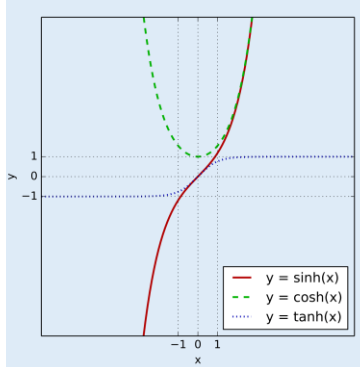
- Activation function is a mathematical processing of amplification or dumping of a sum of weighted neuron's inputs.

- Frequently, artificial neural networks use one of the following activation functions:

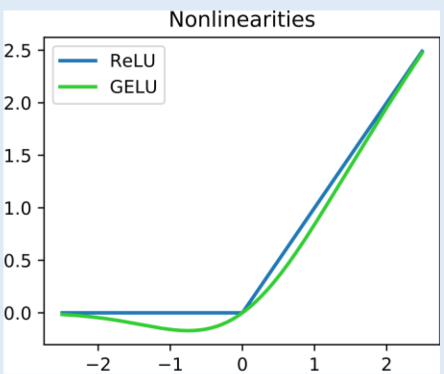
Sigmoid



Tanh



ReLU / GELU



Softmax

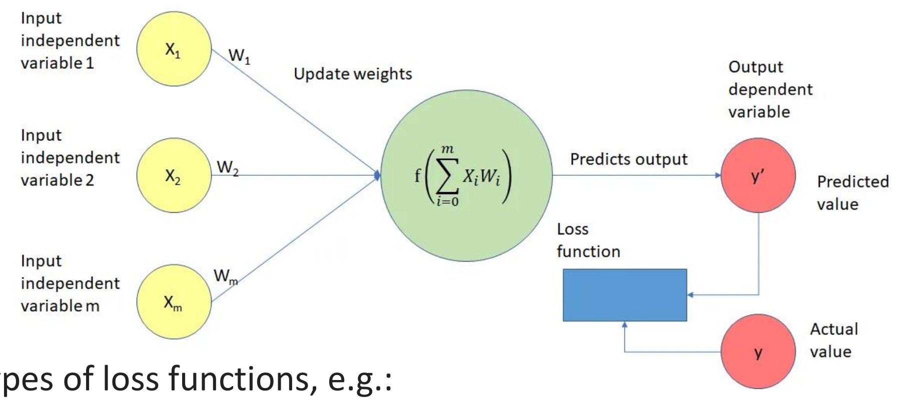
$$p_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Images source: Wikipedia (1-3), Botpenguin.com (4)

University of Essex Online

Model details: loss function

- The loss function represents a difference between calculated and expected output. It helds an important role both in indicating the accuracy, and also in the backpropagation process.
  Hence, its computational complexity is of extreme importance.



- There are many types of loss functions, e.g.:
  - $\log_2$ loss function,
  - Mean Squared Error
  - Mean Absolute Error

The loss function for models with softmax activation function is:

$$L_i = -\log_2 p_i \, .$$

As described earlier, softmax function outcomes are possibilities (here: p_i). Low loss function means that the model is accurate, accordingly.

Picture author: Mohammed Zeeshan Mulla,
https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de

University of Essex
Online

Strategy of the model's design

			Version					
Layers	*Size*	*Description*	I	II	III	IV	**V**	VI
Input layer	32x32x3	Picture size	32x32x3					
Convolutional 2D no.1	29x29x…	Kernel output (29x29) times numer of filters: act. function:	32 ReLU	32 sigm	32 ReLU	32 ReLU	64 ReLU	64 ReLU
MaxPooling2D no.1	14x14x…	Pooling by 2x2 element	As above					
Convolutional 2D no.2	11x11x…	Kernel output (11x11) times numer of filters: act. function:	32 ReLU	32 sigm	32 sigm	64 ReLU	128 ReLU	128 ReLU
MaxPooling2D no.2	5x5x…	Pooling by 2x2 element	As above					
Flattening	800	Flattening the 3D array into 1D	800	800	800	1600	3200	3200
Hidden neural layer	… neurons	No. of network neurons: act. function:	256 ReLU	256 sigm	256 ReLU	256 ReLU	256 ReLU	512 ReLU
Output neural layer	10 neurons	No. of network neurons: act. function:	10 Softmax					
		Model accuracy:	0.636	0.608	0.631	0.683	**0.687**	0.683
		Epochs:	5	17	12	7	**6**	5

University of Essex

Online

Conclusion and outcomes

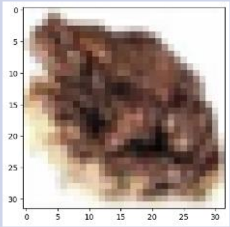
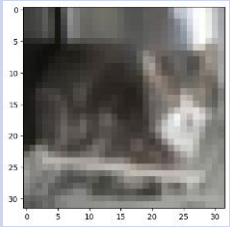
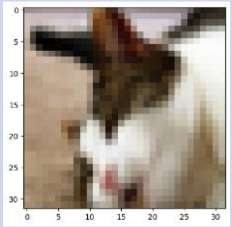
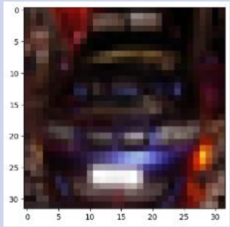
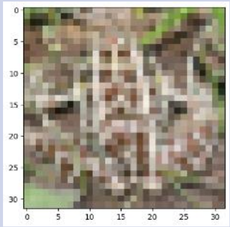
Given network specification

Layers	*Size*
Input layer	32x32x3
Convolutional 2D no.1	29x29x**64**
MaxPooling2D no.1	14x14x64
Convolutional 2D no.2	11x11x**128**
MaxPooling2D no.2	5x5x128
Flattening	800
Hidden neural layer	**256** neurons
Output neural layer	10 neurons

Network performance metrics

- Accuracy: **0.687**
- Precision: 0.67
- Recall: 0.66
- F1-score: 0.66
- Confusion matrix – see Appendix 1

Classification for 5 random elements from validation set

Image					
Original label	frog	cat	cat	automobile	frog
Prediction	frog	cat	deer	automobile	frog

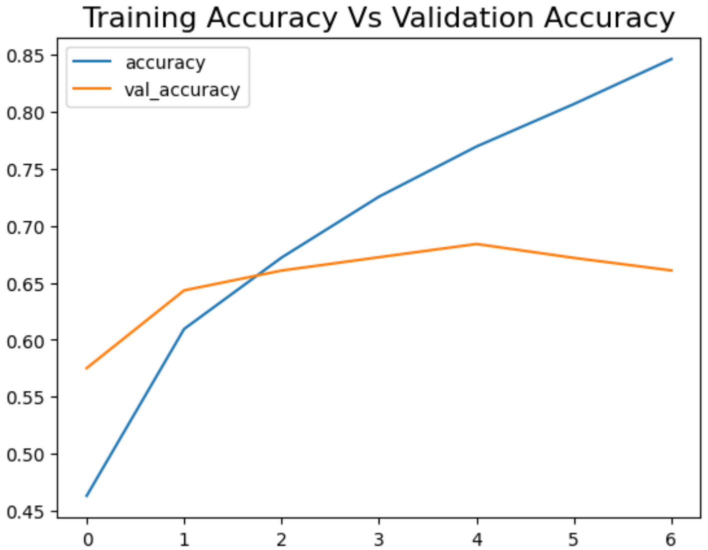
Reflections and lessons learnt

Reflections on the learnings acquired throughout the activity

The importance of profound familiarity with field research prior to constructing the model structure

Proficiency in understanding/sensing when to stop searching for a more optimal model is critical

The crucial role of the validation set



The complementary notions of Recall and Precision

University of Essex Online

References

Bhatt, A. (2024) *Activation Functions, Global Average Pooling, Softmax, Negative Likelihood Loss.* Available at: https://medium.com/@anilaknb/activation-functions-global-average-pooling-softmax-negative-likelihood-loss-86fb50232459 (Accessed: 20.01.2025).

Brownlee, J. (2019) *Loss and Loss Functions for Training Deep Learning Neural Networks.* Available at: *https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/* (Accessed: 20.01.2025).

Kubat, M. (2015) *An introduction to Machine Learning.* Miami: Springer International Publishing.

Pant, A. (2019) *Introduction to Machine Learning for Beginners*. Available at: https://towardsdatascience.com/introduction-to-machine-learning-for-beginners-eed6024fdb08 (Accessed: 20.01.2025).

University of Essex

Online

Appendices

Appendix 1. Confusion matrix

775	29	80	6	8	2	6	7	37	50
15	829	19	8	2	3	5	1	15	103
56	13	733	21	49	42	30	28	11	17
25	23	177	392	60	158	63	28	12	62
27	12	202	34	524	40	45	82	13	21
16	13	169	127	29	553	16	39	8	30
6	20	135	37	39	31	686	8	11	27
20	6	112	24	45	62	8	672	3	48
87	58	21	10	7	8	4	7	737	61
31	100	20	9	1	8	4	5	15	807

Appendix 2. Code listing (1/4)

```
import keras
from keras.datasets import cifar10 as cifar10
import pandas as pd
import numpy as np
from collections import Counter
import tensorflow as tf
from keras.models import Sequential      #to define model/ layers
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten
from sklearn.metrics import confusion_matrix
from IPython.display import display
from keras.preprocessing.image import array_to_img
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

(x_train, label_train), (x_test, label_test) = cifar10.load_data()

print('Shape of a training elements set: '+str(x_train.shape))
print('Shape of a training label set: '+str(label_train.shape))
print('Shape of a testing elements set: '+str(x_test.shape))
print('Shape of a testing label set: '+str(label_test.shape)+'\n\n')

print('Single element dimensions: {} height x {} width x {} colours'.format(x_train.shape[1], x_train.shape[2], x_train.shape[3]))
print('Number of training examples: {}, number of testing examples: {}\n\n'.format(x_train.shape[0], x_test.shape[0]))

LABEL_NAMES = ['airplane', 'automobile','bird','cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

Appendix 2. Code listing (2/4)

```
ind = np.random.randint(0,50000)
x_train[ind]
pic = array_to_img(x_train[ind])
display(pic)

print(LABEL_NAMES[label_train[ind][0]])
print(ind)

unique, counts = np.unique(label_train, return_counts=True)
print(np.asarray((unique, counts)).T)

unique, counts = np.unique(label_test, return_counts=True)
print(np.asarray((unique, counts)).T)

# PREPROCESSING THE DATA
#   NORMALIZING THE DATA
x_train = x_train/255
x_test = x_test/255

VALIDATION_SIZE = 10000
x_valid = x_train[:VALIDATION_SIZE]
label_valid = label_train[:VALIDATION_SIZE]
x_train = x_train[VALIDATION_SIZE:]
label_train = label_train[VALIDATION_SIZE:]

print('Values (validation set): {} Categories (validation set): {} '.format(x_valid.shape, label_valid.shape))
print('Values (train set): {} Categories (train set): {} '.format(x_train.shape, label_train.shape))
print('Values (test set): {} Categories (test set): {} '.format(x_test.shape, label_test.shape))
```

Appendix 2. Code listing (3/4)

```
label_valid_cat = to_categorical(label_valid,num_classes=10)
label_train_cat = to_categorical(label_train,num_classes=10)
label_test_cat = to_categorical(label_test,num_classes=10)

# BUILDING THE MODEL
model = Sequential()

## ************* FIRST SET OF LAYERS ************************
# CONVOLUTIONAL LAYER
model.add(Conv2D(filters=64, kernel_size=(4,4),input_shape=(32, 32, 3), activation='relu',))

# POOLING LAYER
model.add(MaxPool2D(pool_size=(2, 2)))

## *************** SECOND SET OF LAYERS **********************
# Since the shape of the data is 32 x 32 x 3 =3072 ...
# We need to deal with this more complex structure by adding yet another convolutional layer
# *************CONVOLUTIONAL LAYER
model.add(Conv2D(filters=128, kernel_size=(4,4),input_shape=(32, 32, 3), activation='relu',))

# POOLING LAYER
model.add(MaxPool2D(pool_size=(2, 2)))

# FLATTEN IMAGES FROM 32 x 32 x 3 =3072 BEFORE FINAL LAYER
model.add(Flatten())

# 256 NEURONS IN DENSE HIDDEN LAYER (YOU CAN CHANGE THIS NUMBER OF NEURONS)
model.add(Dense(256, activation='relu'))
```

Appendix 2. Code listing (4/4)

```
# LAST LAYER IS THE CLASSIFIER, THUS 10 POSSIBLE CLASSES
model.add(Dense(10, activation='softmax'))
model.summary()

model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])

from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss',patience=2)

history = model.fit(x_train,label_train_cat,epochs=25,validation_data=(x_valid,label_valid_cat),callbacks=[early_stop])
metrics = pd.DataFrame(model.history.history)

metrics[['accuracy', 'val_accuracy']].plot()
plt.title('Training Accuracy Vs Validation Accuracy', fontsize=16)
plt.show()

from sklearn.metrics import classification_report, confusion_matrix
predictions = np.argmax(model.predict(x_test), axis=-1)
print(classification_report(label_test,predictions))
rand = np.random.randint(0,len(x_valid))
plt.imshow(x_valid[rand])

print(np.argmax(model.predict(x_valid[rand].reshape(1,32,32,3))))
print('Predicted: '+LABEL_NAMES[np.argmax(model.predict(x_valid[rand].reshape(1,32,32,3)))])
print('Original label: '+LABEL_NAMES[label_valid[rand][0]])

confusion_matrix(label_test,predictions)
```