Examen Deep Learning

- Naam: Floris Buyse
- Klas: G3C1

Bij dit examen mag je enkel de volgende hulpmiddelen gebruiken

- Google Colab
- De websites Keras en Tensorflow en alle deelpagina's daarop.
- De slides, oefeningen (en oplossingen) zoals ze op Chamilo zijn verschenen.
- Alle zelfgemaakte oefeningen en notities.
- Het handboek "Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow"

Indienen gebeurt door dit notebook uit te voeren en met de uitvoer intact op te laden in de Examen-opdracht. Laad ook een PDF-versie van je notebook op in de examentool.

Alle code die je moet schrijven moet je invullen tussen `### JOUW CODE HIER` en `### EINDE JOUW CODE HIER`.

Vergeet ook niet de korte vraagjes op het antwoordblad in te vullen.

Schrijf al je import statements in de eerste cel hieronder.

```
import tensorflow as tf
import pandas as pd
import numpy as np
### JOUW CODE HIER
from functools import partial
### EINDE JOUW CODE HIER
```

Vraag 1: Wijnkwaliteit

(a) Dataset downloaden.

```
URL = "https://archive.ics.uci.edu/static/public/186/wine+quality.zip"
### JOUW CODE HIER
ds = tf.keras.utils.get_file(origin=URL, extract=True, cache_dir=".")
### EINDE JOUW CODE HIER
```

```
Downloading data from https://archive.ics.uci.edu/static/public/186/wine+quality.zip
   8192/Unknown - 0s 0us/step
```

(b) Inlezen in pandas dataframe. Verifieer dat de dataset 4898 rijen en 12 kolommen heeft.

```
file_path = "./datasets/winequality-white.csv" # Pas aan!
wine_df = pd.read_csv(file_path, delimiter=";")
wine_raw_data = wine_df.values
wine_df.shape
```

```
(4898, 12)
```

```
wine_df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

(c) Opsplitsen in trainings en validatiedata.

```
### JOUW CODE HIER
wine_raw_data_train = wine_raw_data[:4000]
wine_raw_data_val = wine_raw_data[4000:]
wine_raw_data_train.shape, wine_raw_data_val.shape
### EINDE JOUW CODE HIER
```

```
((4000, 12), (898, 12))
```

(d) Maak features en labels.

```
### JOUW CODE HIER
X_train = wine_raw_data_train[:, :-1]
X_val = wine_raw_data_val[:, :-1]
y_train = wine_raw_data_train[:, -1]
y_val = wine_raw_data_val[:, -1]

X_train.shape, y_train.shape
### EINDE JOUW CODE HIER
```

```
((4000, 11), (4000,))
```

(e) Maak Keras laag.

```
### JOUW CODE HIER
preprocess_layer = tf.keras.layers.Normalization()
preprocess_layer.adapt(X_train)
### EINDE JOUW CODE HIER
```

(f) Gemiddelde en standaardafwijking.

```
### JOUW CODE HIER
mu = tf.math.reduce_mean(X_val, axis=0)
sigma = tf.math.reduce_std(X_val, axis=0)
mu, sigma
### EINDE JOUW CODE HIER
```

```
(<tf.Tensor: shape=(11,), dtype=float64, numpy=
 array([6.39855234e+00, 2.89036748e-01, 3.03429844e-01, 6.51325167e+00,
        4.64042316e-02, 3.41236080e+01, 1.28881960e+02, 9.93312578e-01,
        3.17178174e+00, 4.92561247e-01, 1.07505345e+01])>,
 <tf.Tensor: shape=(11,), dtype=float64, numpy=
 array([6.71971568e-01, 1.02778178e-01, 1.08764248e-01, 4.98213538e+00,
        2.13303182e-02, 1.73817603e+01, 3.74626390e+01, 2.87669270e-03,
        1.41074264e-01, 1.11509730e-01, 1.26889753e+00])>)
```

(g) Definieer `get_model`.

```
X_train[0].shape
```

```
(11,)
```

```
### JOUW CODE HIER
def get_model(units, preprocess_layer=None, activation=None):
  model = tf.keras.Sequential()
  model.add(tf.keras.layers.Input(shape=(11,)))

  if preprocess_layer != None:
    model.add(preprocess_layer)

  model.add(tf.keras.layers.Dense(20, activation="relu"))

  if activation is None:
    model.add(tf.keras.layers.Dense(units))
  else:
    model.add(tf.keras.layers.Dense(units, activation=activation))


  return model
### EINDE JOUW CODE HIER
```

(h) Roep `get_model` aan voor een classificatieprobleem.

```
### JOUW CODE HIER
model = get_model(10, preprocess_layer, "softmax")
model.summary()
### EINDE JOUW CODE HIER
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_2 (Normaliza  (None, 11)                23
 tion)

 dense_4 (Dense)             (None, 20)                240

 dense_5 (Dense)             (None, 10)                210

=================================================================
Total params: 473 (1.85 KB)
Trainable params: 450 (1.76 KB)
Non-trainable params: 23 (96.00 Byte)
_________________________________________________________________
```

(i) Compileer model.

```
### JOUW CODE HIER
model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.0005), metrics=["accuracy"], loss="sparse_categorical_crossentropy"
### EINDE JOUW CODE HIER
```

(j) Train model.

```
### JOUW CODE HIER
model_check = tf.keras.callbacks.ModelCheckpoint("model.tf", save_best_only=True, monitor="val_accuracy")
escb = tf.keras.callbacks.EarlyStopping(restore_best_weights=False, monitor="accuracy", min_delta=0.1, patience=3)
model.fit(X_train, y_train, batch_size=128, epochs=100, validation_data=(X_val, y_val), callbacks=[escb, model_check])
### EINDE JOUW CODE HIER
```

```
Epoch 1/100
32/32 [==============================] - 1s 30ms/step - loss: 1.3221 - accuracy: 0.4773 - val_loss: 1.2554 - val_accuracy: 0.4911
Epoch 2/100
32/32 [==============================] - 0s 3ms/step - loss: 1.2962 - accuracy: 0.4855 - val_loss: 1.2343 - val_accuracy: 0.4866
Epoch 3/100
32/32 [==============================] - 0s 4ms/step - loss: 1.2744 - accuracy: 0.4902 - val_loss: 1.2154 - val_accuracy: 0.4900
Epoch 4/100
32/32 [==============================] - 0s 4ms/step - loss: 1.2558 - accuracy: 0.4985 - val_loss: 1.2029 - val_accuracy: 0.4833
<keras.src.callbacks.History at 0x7e60103bcfa0>
```

(k) Evalueer beste model op validatiedata.

```
### JOUW CODE HIER
model = tf.keras.models.load_model("model.tf")
model.evaluate(X_val, y_val)
### EINDE JOUW CODE HIER
```

```
29/29 [==============================] - 0s 2ms/step - loss: 1.2554 - accuracy: 0.4911
[1.2554370164871216, 0.4910913109779358]
```

(l) Roep `get_model` aan voor een regressieprobleem.

```
### JOUW CODE HIER
model2 = get_model(1, preprocess_layer)
model2.summary()
### EINDE JOUW CODE HIER
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_2 (Normaliza  (None, 11)                23
 tion)

 dense_6 (Dense)             (None, 20)                240

 dense_7 (Dense)             (None, 1)                 21

=================================================================
Total params: 284 (1.11 KB)
Trainable params: 261 (1.02 KB)
Non-trainable params: 23 (96.00 Byte)
_________________________________________________________________
```

(m) Compileer en train tweede model.

```
### JOUW CODE HIER
model.compile(optimizer="adam", loss="mse", metrics=["mse"])
model_check = tf.keras.callbacks.ModelCheckpoint("model2.tf", save_best_only=True, monitor="val_loss")
model.fit(X_train, y_train, batch_size=128, epochs=100, validation_data=(X_val, y_val), callbacks=[model_check])
### EINDE JOUW CODE HIER
```

```
Epoch 29/100
32/32 [==============================] - 0s 3ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 30/100
32/32 [==============================] - 0s 4ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 31/100
32/32 [==============================] - 0s 4ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 32/100
32/32 [==============================] - 0s 3ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 33/100
32/32 [==============================] - 0s 4ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 34/100
32/32 [==============================] - 0s 4ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 35/100
32/32 [==============================] - 0s 3ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 36/100
32/32 [==============================] - 0s 4ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 37/100
32/32 [==============================] - 0s 5ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 38/100
32/32 [==============================] - 0s 7ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 39/100
32/32 [==============================] - 0s 7ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 40/100
32/32 [==============================] - 0s 6ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 41/100
32/32 [==============================] - 0s 5ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 42/100
32/32 [==============================] - 0s 6ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 43/100
32/32 [==============================] - 0s 6ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 44/100
32/32 [==============================] - 0s 8ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 45/100
32/32 [==============================] - 0s 7ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 46/100
32/32 [==============================] - 0s 6ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 47/100
32/32 [==============================] - 0s 8ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 48/100
32/32 [==============================] - 0s 5ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 49/100
32/32 [==============================] - 0s 7ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 50/100
32/32 [==============================] - 0s 5ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 51/100
32/32 [==============================] - 0s 5ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 52/100
32/32 [==============================] - 0s 5ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 53/100
32/32 [==============================] - 0s 6ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 54/100
32/32 [==============================] - 0s 3ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 55/100
32/32 [==============================] - 0s 3ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 56/100
32/32 [==============================] - 0s 4ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 57/100
32/32 [==============================] - 0s 4ms/step - loss: 34.2333 - mse: 34.2333 - val_loss: 33.8795 - val_mse: 33.8795
Epoch 58/100
```

(n) Accuraatheid tweede model.

```
model2.predict(X_val).shape
```

```
29/29 [==============================] - 0s 3ms/step
(898, 1)
```

```
def accuracy(labels, raw_predictions):
    ### JOUW CODE HIER
    total_error = 0.0
    total_samples = 0
    for item in raw_predictions:
        labels = labels * sigma[-1] + mu[-1]
        total_error += np.sum(np.abs(labels - item))
        total_samples += item.shape[0]

    return total_error / total_samples

print(accuracy(y_val, model2.predict(X_val)))
    ### EINDE JOUW CODE HIER
```

```
 29/29 [==============================] - 0s 2ms/step
 3.704559160103956e+78
```

Vraag 2: Xception

(a) Schrijf hieronder code voor `DefaultSeparableConv`.

```
### JOUW CODE HIER
DefaultSeparableConv = partial(tf.keras.layers.SeparableConv2D,
                               kernel_size=(3,3))
### EINDE JOUW CODE HIER
```

(b) Schrijf hieronder code voor `XceptionModule`

```
### JOUW CODE HIER
class XceptionModule(tf.keras.layers.Layer):
  def __init__(self, filters, use_max_pool=False, **kwargs):
        super().__init__(**kwargs)

        main_path = []
        skip_path = []

        self.use_max_pool = use_max_pool
        self.conv1 = DefaultSeparableConv(filters=256)
        self.conv2 = DefaultSeparableConv(filters=728)
        self.activation = tf.keras.layers.ReLU()
        self.maxpool =  tf.keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2))
        self.conv_skip = tf.keras.layers.Conv2D(filters, kernel_size=(1, 1), strides=(2,2))

  def call(self, inputs):

    if self.use_max_pool:
      skip = inputs
      inputs = self.maxpool(inputs)
      inputs = self.conv1(inputs)
      inputs = self.activation(inputs)
      inputs = self.conv1(inputs)
      inputs = self.activation(inputs)

      skip = self.conv_skip(skip)
      inputs = tf.keras.layers.Add()[skip, inputs]
      return inputs

    else:
      skip = inputs
      inputs = self.conv2(inputs)
      inputs = self.activation(inputs)
      inputs = self.conv2(inputs)
      inputs = self.activation(inputs)
      inputs = self.conv2(inputs)
      inputs = self.activation(inputs)

      inputs = tf.keras.layers.Add()[skip, inputs]
      return inputs
### EINDE JOUW CODE HIER
```

(c) Hieronder kan je code schrijven om de kleine vraagjes op het antwoordblad op te lossen. Deze code wordt als dusdanig niet gequoteerd.

```
### JOUW CODE HIER
xception = XceptionModule(filters=64, use_max_pool=False)
xception2 = XceptionModule(filters=64, use_max_pool=True)

X = tf.random.uniform(shape=(2, 100, 100, 64))
xception(X)
### EINDE JOUW CODE HIER
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-177-7ea5e677993d> in <cell line: 6>()
      4
      5 X = tf.random.uniform(shape=(2, 100, 100, 64))
----> 6 xception(X)
      7 ### EINDE JOUW CODE HIER

 1 frames
<ipython-input-66-55f14c74b568> in call(self, inputs)
     32       inputs = self.conv2(inputs)
     33       inputs = self.activation(inputs)
---> 34       inputs = self.conv2(inputs)
     35       inputs = self.activation(inputs)
     36       inputs = self.conv2(inputs)

ValueError: Exception encountered when calling layer 'xception_module_7' (type XceptionModule).

Input 0 of layer "separable_conv2d_15" is incompatible with the layer: expected axis -1 of input shape to have value 64, but
received input with shape (2, 98, 98, 728)

Call arguments received by layer 'xception_module_7' (type XceptionModule):
  • inputs=tf.Tensor(shape=(2, 100, 100, 64), dtype=float32)
```

SEARCH STACK OVERFLOW

Vraag 3: Jena Dataset

```
# Deze code maakt een numpy-array "raw_data"
FILE_PATH = "./jena_climate_2009_2016.csv"
raw_data = pd.read_csv(FILE_PATH).drop("Date Time", axis="columns").values
raw_data.shape # Moet printen (420451, 14)
```

```
(225064, 14)
```

```
# Deze code creëert een Dataset bestaande uit tupels
def create_ds(raw_data, target):
    sampling_rate=6
    sequence_length=120
    delay=sampling_rate * (sequence_length + 24 -1)
    return tf.keras.utils.timeseries_dataset_from_array(
        data=raw_data,
        targets=target[delay:],
        sampling_rate=sampling_rate,
        sequence_length=sequence_length,
        shuffle=False,
        batch_size=None
    )
```

```
# Voorbeeld met énkel indices
example_data = np.arange(2000)
example_ds = create_ds(example_data, example_data)
for features, target in example_ds.take(3):
    print(f"{features}")
    print(features.shape)
    print(f"{target}")
    print("*"*50)
```

```
[  0   6  12  18  24  30  36  42  48  54  60  66  72  78  84  90  96 102
 108 114 120 126 132 138 144 150 156 162 168 174 180 186 192 198 204 210
 216 222 228 234 240 246 252 258 264 270 276 282 288 294 300 306 312 318
 324 330 336 342 348 354 360 366 372 378 384 390 396 402 408 414 420 426
 432 438 444 450 456 462 468 474 480 486 492 498 504 510 516 522 528 534
 540 546 552 558 564 570 576 582 588 594 600 606 612 618 624 630 636 642
 648 654 660 666 672 678 684 690 696 702 708 714]
(120,)
858
**************************************************
[  1   7  13  19  25  31  37  43  49  55  61  67  73  79  85  91  97 103
 109 115 121 127 133 139 145 151 157 163 169 175 181 187 193 199 205 211
 217 223 229 235 241 247 253 259 265 271 277 283 289 295 301 307 313 319
 325 331 337 343 349 355 361 367 373 379 385 391 397 403 409 415 421 427
```

```
 433 439 445 451 457 463 469 475 481 487 493 499 505 511 517 523 529 535
 541 547 553 559 565 571 577 583 589 595 601 607 613 619 625 631 637 643
 649 655 661 667 673 679 685 691 697 703 709 715]
(120,)
859
**************************************************
[  2   8  14  20  26  32  38  44  50  56  62  68  74  80  86  92  98 104
 110 116 122 128 134 140 146 152 158 164 170 176 182 188 194 200 206 212
 218 224 230 236 242 248 254 260 266 272 278 284 290 296 302 308 314 320
 326 332 338 344 350 356 362 368 374 380 386 392 398 404 410 416 422 428
 434 440 446 452 458 464 470 476 482 488 494 500 506 512 518 524 530 536
 542 548 554 560 566 572 578 584 590 596 602 608 614 620 626 632 638 644
 650 656 662 668 674 680 686 692 698 704 710 716]
(120,)
860
**************************************************
```

(a) Vervolledig de methode `create_ds_with_window`.

```
def create_ds_with_window(raw_data):
    ds = tf.data.Dataset.from_tensor_slices(raw_data)
    return (ds.
            window(size=120+24, shift=1, stride=6, drop_remainder=True)
            .flat_map(lambda x: x.batch(120+24))
            .map(lambda x: (x[:120], x[-1, 1]))
           )


# Deze code controleert je antwoord op de vorige vraag
for item1, item2 in zip(create_ds_with_window(raw_data).take(10),
                        create_ds(raw_data, target=raw_data[:, 1]).take(10)):
    assert np.allclose(item1[0], item2[0]), "Sequenties niet gelijk"
    assert np.allclose(item1[1], item2[1]), "Targets niet gelijk"

print("Alle testen geslaagd")
```

```
Alle testen geslaagd
```

(b) Vervolledig de methode `create_seq_to_seq_ds_aux`.

```
def create_seq_to_seq_ds_aux(dataset):
    return (dataset.
            window(size=120+24, shift=1, stride=6)
            .flat_map(lambda x: x.batch(120+24))
            .window(size=120, shift=1)
            .flat_map(lambda x: x.batch(2))
            .map(lambda x: (x[0][:120], x[1][24:]))
           )


# Test code
for item, target in create_seq_to_seq_ds_aux(tf.data.Dataset.from_tensor_slices(raw_data)).take(3):
  assert item.shape == (120, 14)
  assert target.shape == (120,)

print("Alle testen geslaagd. De shapes zijn correct")
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-235-c1a9be82ebdd> in <cell line: 2>()
      2 for item, target in create_seq_to_seq_ds_aux(tf.data.Dataset.from_tensor_slices(raw_data)).take(3):
      3   assert item.shape == (120, 14)
----> 4   assert target.shape == (120,)
      5
      6 print("Alle testen geslaagd. De shapes zijn correct")

AssertionError:
```

SEARCH STACK OVERFLOW

(c) Vervolledig de methode `create_seq_to_seq_ds`.

```
def create_seq_to_seq_ds(raw_data):
    ds = tf.data.Dataset.from_tensor_slices(raw_data)
    return tf.data.Dataset.range(6).interleave(
        lambda i : tf.data.TextLineDataset(i))
```