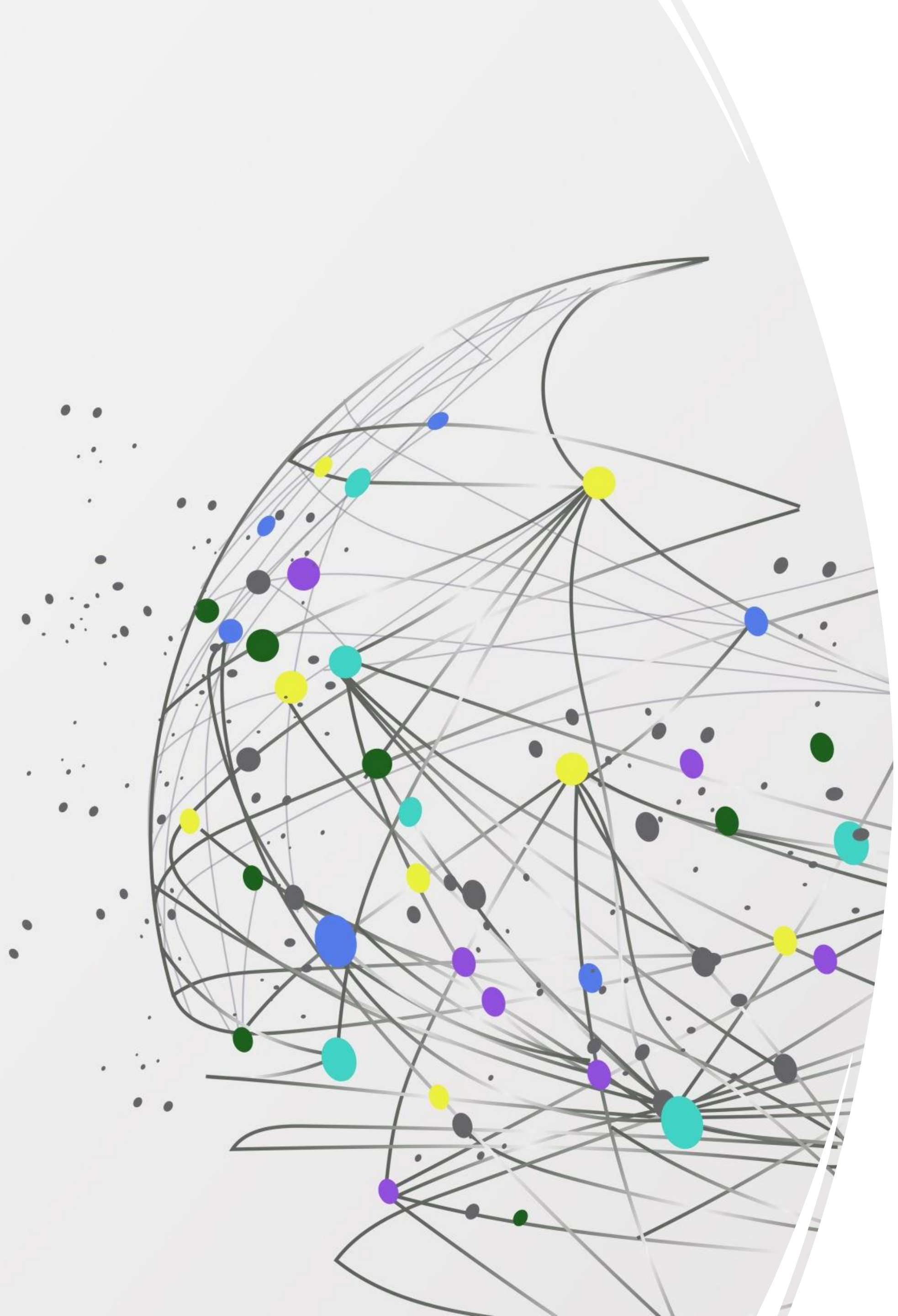


DSI DEEP LEARNING SHORT COURSE

Practical Introduction to CNNs and RNNs with Keras

25 June 2024

Prof Mandla Gwetu [mgwetu@sun.ac.za]

Dr Gabriel Kallah-Dagadu [KallahDagaduG@ukzn.ac.za]

Dr Mohanad Mohammed [MohammedM1@ukzn.ac.za]

PRESENTATION OUTLINE

- Learning outcomes
- Theoretical concepts
- Keras functional API
- Project

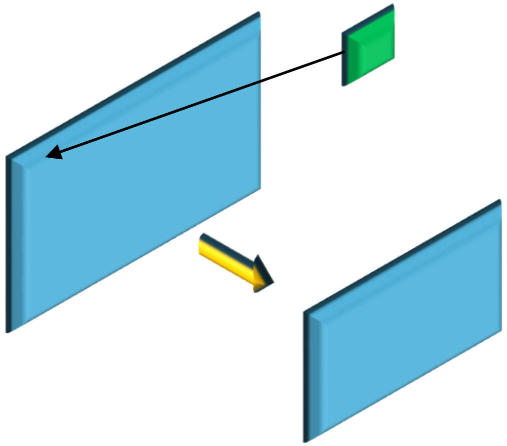
LEARNING OUTCOMES

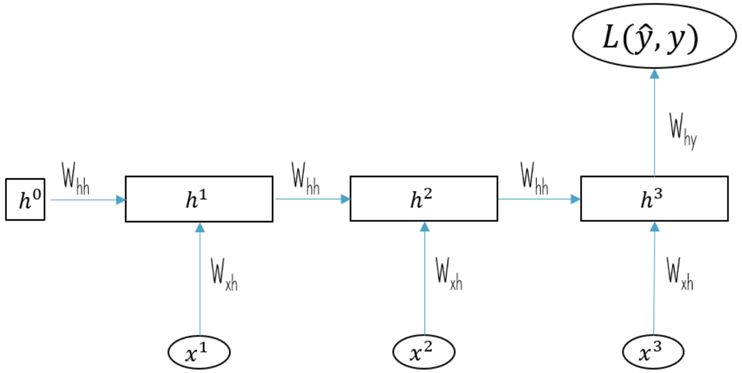
- The following learning outcomes are expected at the end of this session:
  - Correctly describe the Keras functional API and identify use cases for which it is ideal.
  - Accurately interpret the CNN and RNN architecture diagrams and explain their model functionality.

THEORETICAL CONCEPTS

- CNNs process image input using a filter/kernel window that slides over similarly sized tiles of the input image while generating pixel values for an output image. Element-wise multiplication between the input image tile and kernel pixel values is followed by summation operation, bias addition and activation function transformation.
- The output window dimensions is calculated as:

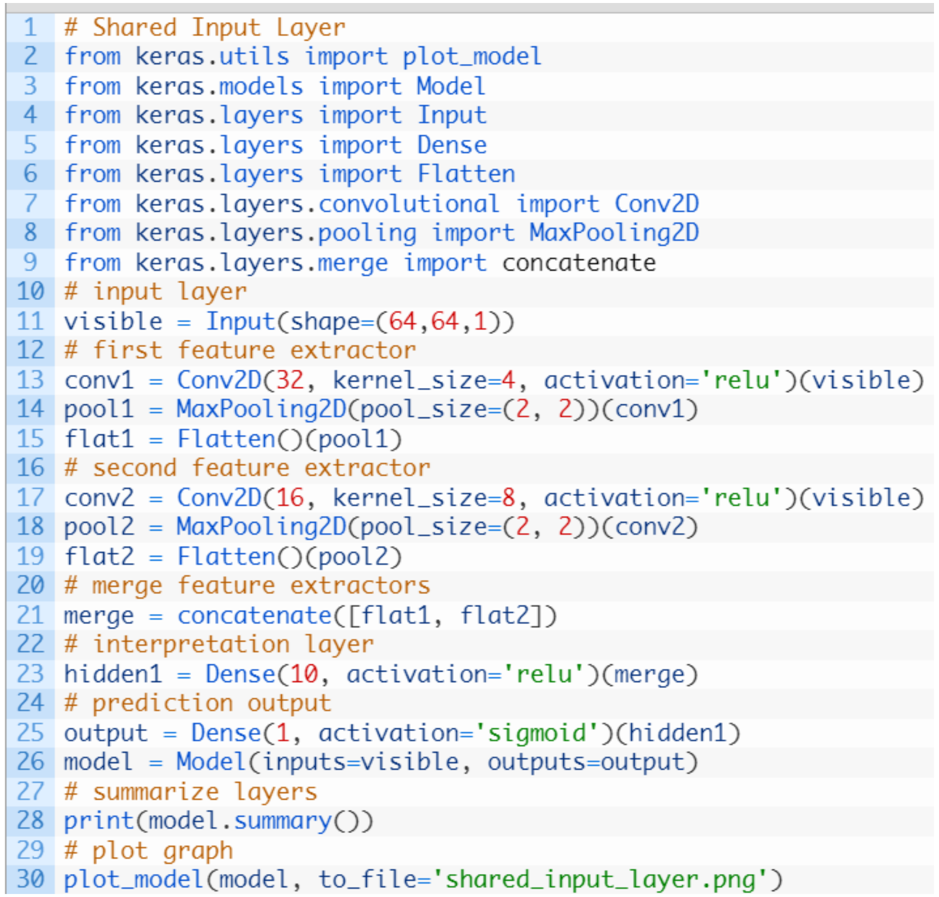
  $O = \frac{(I-F+2P)}{S} + 1$, where I is the input image dimension, F is the filter size, P is the padding and S is the step size (stride).
- RNNs process temporal data whose instances are organized as timestamp snapshots of features. RNNs maintain a hidden state that captures patterns within inputs over time and is represented as: h^t = $tanh(W^T{}_{hh}h^{t-1} + W^T{}_{xh}x^t)$, where the initial hidden state h^0 is a constant..
- When rolled out, RNNs can form one-to-one, one-to-many, many-to-one or many-to-many structures based on the synchronization between their input and output.



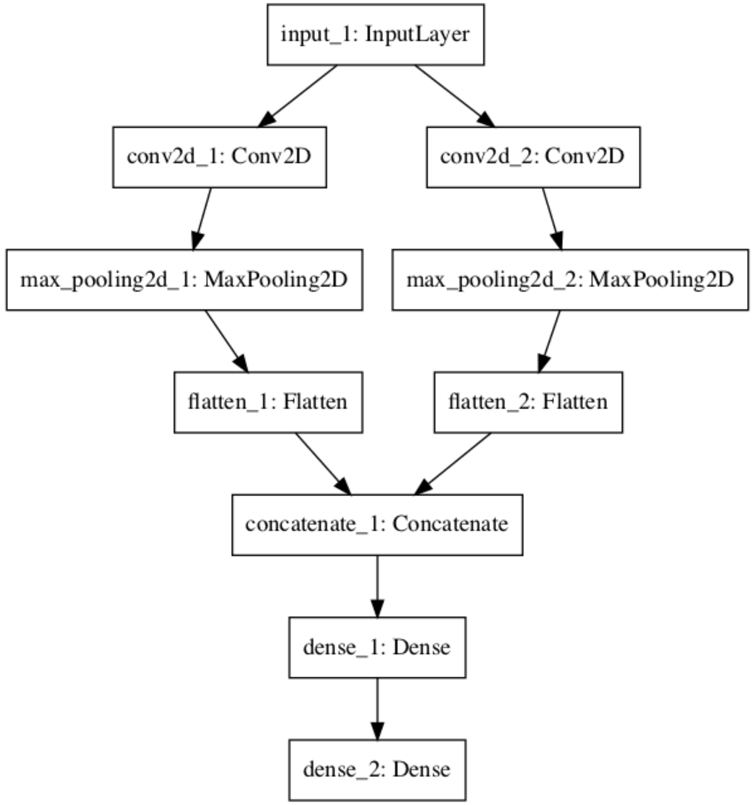


KERAS FUNCTIONAL API

The Keras functional API enables the construction of specialized model architectures which have properties such as shared inputs, shared layers, multiple inputs and multiple outputs.



```python
# Shared Input Layer
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
# input layer
visible = Input(shape=(64,64,1))
# first feature extractor
conv1 = Conv2D(32, kernel_size=4, activation='relu')(visible)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
flat1 = Flatten()(pool1)
# second feature extractor
conv2 = Conv2D(16, kernel_size=8, activation='relu')(visible)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
flat2 = Flatten()(pool2)
# merge feature extractors
merge = concatenate([flat1, flat2])
# interpretation layer
hidden1 = Dense(10, activation='relu')(merge)
# prediction output
output = Dense(1, activation='sigmoid')(hidden1)
model = Model(inputs=visible, outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='shared_input_layer.png')
```



Source: How to Use the Keras Functional API for Deep Learning – MachineLearningMastery.com

DNN-CNN-RNN IMPLEMENTATION PROJECT

- Select or create a dataset on which you can apply a DNN, CNN, RNN, or combination them towards a prediction task.
- Tackle this project in groups of not more than 6 individuals.
- You will be required to make a 20-minute (followed by 10 minutes of questions) proposal presentation about your proposed project on Wednesday.
- The proposal presentation should explain the problem to be solved, propose a solution, explain the methods it uses and justify its suitability for the problem then present preliminary work towards the solution.
- All group members should participate in the presentation.
- Final presentations will be presented on Friday and should demonstrate knowledge of the typical deep learning data pipeline.