Class 4

Power System Event Classification with Convolutional Neural Network Using PMU Data – Model Assessment

Outline

- Revisit saving and loading models.
- Hyper-parameter tuning with validation dataset.
- Visualization of accuracy and loss during training process.
- Assignment walk-through.

Outline

- Revisit saving and loading models.
- Hyper-parameter tuning with validation dataset.
- Visualization of accuracy and loss during training process.
- Assignment walk-through.

Revisit Saving and Loading Models

- Saving models requires to install h5py library. It is usually installed as a dependency with TensorFlow. If it is not installed, you can install it in the Command Prompt with the command below.

“sudo pip install h5py”

- To save a trained model, simply use the “.save(directory)” command in Python.

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
# define model
model = Sequential()
model.add(Dense(10, input_dim=5, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='linear'))
# Compile model
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squared_error'])
# Fit the model
model.fit(X, Y, epochs=100, batch_size=10, verbose=0)
# save model and architecture to single file
model.save("SaveFolder/model_name.h5") # the directory and file name of the saved file.
```

Revisit Saving and Loading Models—Continued

- Another way to save a model:

```
# equivalent to: model.save("SaveFolder/model_name.h5")
from tensorflow.keras.models import save_model
save_model(model, "SaveFolder/model_name.h5")
```

- We can load the above saved model using the function “load_model”

```
# load and evaluate a saved model
from tensorflow.keras.models import load_model
# load model
model_from_save = load_model("SaveFolder/model_name.h5")
```

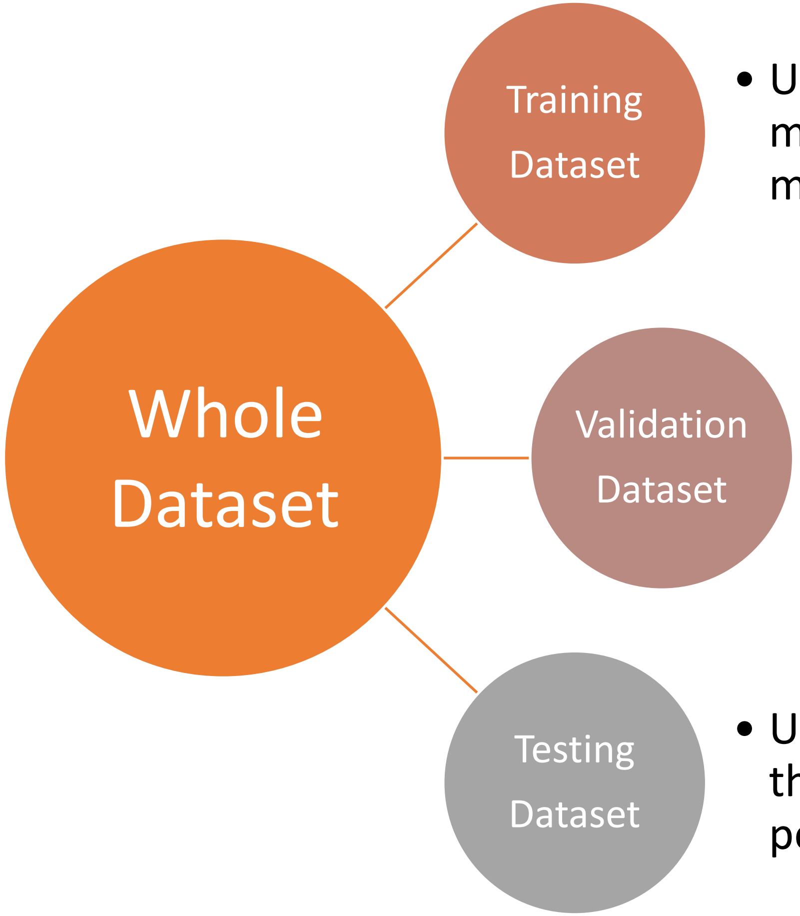
Outline

- Revisit saving and loading models.
- Hyper-parameter tuning with validation dataset.
- Visualization of accuracy and loss during training process.
- Assignment walk-through.

Hyper-Parameter Tuning

- Hyper-parameters are the settings and parameters that control the configuration of machine learning models.
- They cannot be trained during the model training procedure.
- They influence the performance (e.g., accuracy, loss) of the models.
- Examples Hyper-parameters of a Convolutional Neural Network (CNN) model:
  - Batch size.
  - Optimizer algorithm that trains the model.
  - Number of the layer
  - Filter size of each Convolutional layer.

Dataset Split



- Used to train machine learning models.
- Used for hyperparameter tuning and early stopping.
- Used to evaluate the prediction performance.

Note: Do not use testing data for the hyper-parameter tuning!

Hyper-Parameter Tuning with Validation Set

- We split the dataset into three parts: training set, validation set, and testing set.
- We train the model with the training set for each hyper-parameter setup and evaluate its performance with the validation set.
- We evaluate the model performance with the validation set to explore hyper-parameters and assess prediction performance.
- The hyper-parameter set that achieves the highest performance on the validation set is selected as the optimal choice.

Grid Search

- Each hyper-parameter has a list of possible values.
- Evaluate the performance of all the possible combinations (so-called "grid") of hyper-parameter values.
- Example 1: suppose batch size = [8, 16, 32], Optimizer = [SGD, Adam], then the grid has $3 \times 2 = 6$ possible combinations.
- Example 2: if one additional hyper-parameter is the number of the filters in a convolutional layer = [32, 64, 128], then the grid has $3 \times 2 \times 3 = 18$ combinations.

Grid Shape!

Example 1:

Batch size=08, Optimizer=SGD	**Batch size=16, Optimizer=SGD**	**Batch size=32, Optimizer=SGD**
Batch size=08, Optimizer =Adam	**Batch size=16, Optimizer =Adam**	**Batch size=32, Optimizer =Adam**

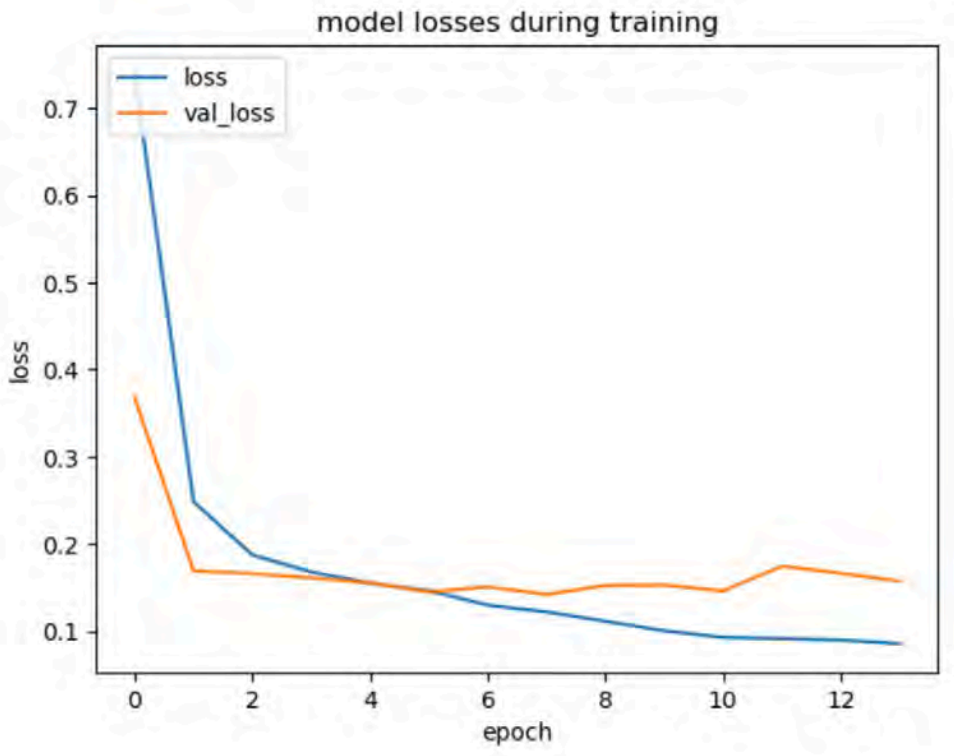
Outline

- Revisit saving and loading models.
- Hyper-parameter tuning with validation dataset.
- Visualization of accuracy and loss during training process.
- Assignment walk-through.

Visualization of loss and accuracy during training

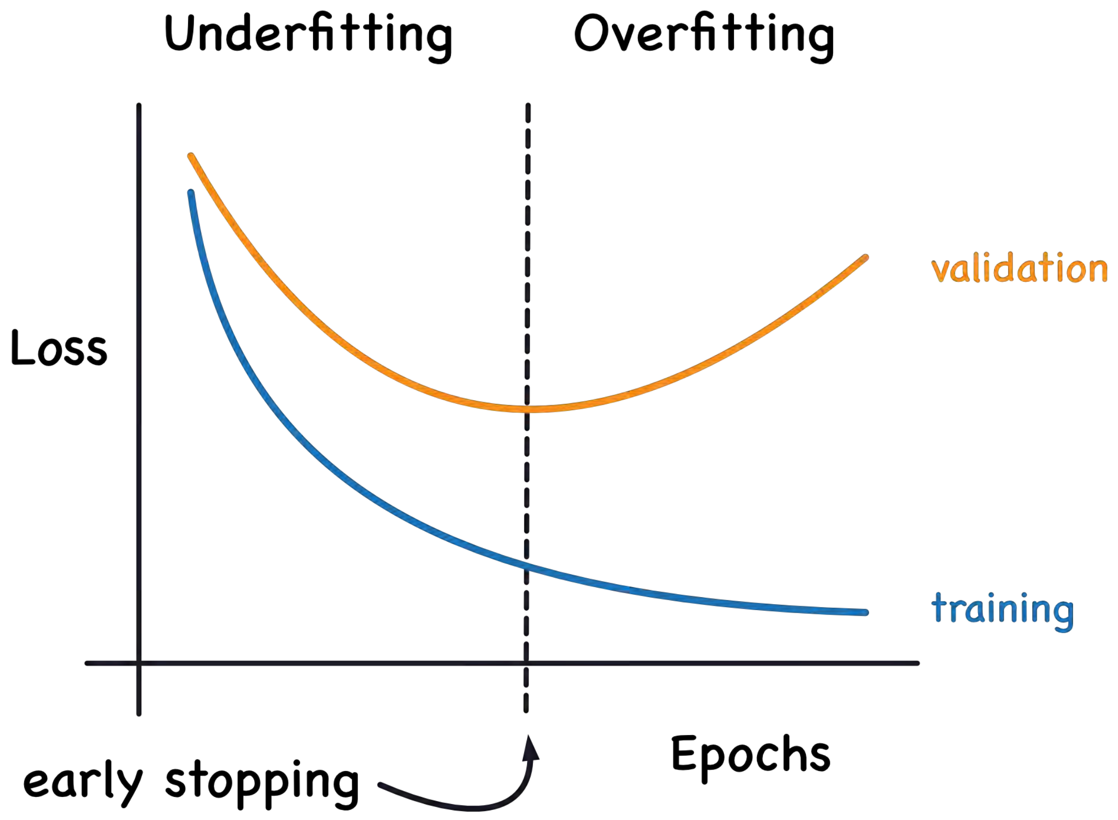
- Plot the loss and accuracy can help monitor the training of the model.
- Example of visualizing losses during the training:

```python
import matplotlib.pyplot as plt

loss = history.history['loss']
val_loss = history.history['val_loss']
plt.plot(loss, label='loss')
plt.plot(val_loss, label='val_loss')
plt.title('model losses during training')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(loc='upper left')
plt.show()
plt.close()
```



Visualization of loss and accuracy during training

- **Underfitting** the training set is when the loss is not as low as it could be because the model hasn't learned enough *signal*.
- **Overfitting** the training set is when the loss is not as low as it could be because the model learned too much *noise*.



Outline

- Revisit Saving and Loading Models
- Hyper-parameter tuning with validation dataset.
- Visualization of accuracy and loss during training process.
- Assignment walk-through

Load Needed Libraries/Packages

- To start, we need to load the libraries and packages that will used in the code.

```
# Load the tensorflow, which is a framework for deep learning.
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
# Load numpy library as "np", which can handle large matrices and provides some mathematical functions.
import numpy as np
# Load pandas as "pd", which is useful when working with data tables.
import pandas as pd
# Load the SciPy, which provides many useful tools for scientific computing
import scipy
# Load random, which provide some randomize functions.
import random
# Load a function pyplot as "plt" to plot figures.
import matplotlib.pyplot as plt
# Load functions to calculate precision, and recall
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Load the VAR function for Vector Autoregression
from statsmodels.tsa.api import VAR

# Setup the random seed for reproducibility
seed = 1234
random.seed(seed)
np.random.seed(seed)
```

Grid Search for Hyper-Parameter Tuning

- In your code, you are required to define and optimize hyperparameters to maximize performance on the validation dataset.
- To achieve this, you should conduct a grid search over the hyperparameter space to systematically explore different settings and identify the optimal configuration.

```
# Further Hyper-parameters


##------------------------------------------------------------------------##
##---------------------Students start filling below----------------------##
##------------------------------------------------------------------------##


"""
    Try different combination of the hyper-parameters.
    Using grid search to get the best hyper-parameters
    Choose the best hyper-parameter combination to train the final model.
    Pick your own hyper-parameters and search range.
"""


##------------------------------------------------------------------------##
##----------------------------End filling---------------------------------##
##------------------------------------------------------------------------##
```

Train the model

- You are required to train your final model using the best hyper-parameters set obtained from the previous step.

```
##------------------------------------------------------------------------##
##--------------------Students start filling below---------------------##
##------------------------------------------------------------------------##

"""
    Using the best hyper-parameters get from the previous step and train the final model.
"""


""" Filling code below """


""" End Filling """

##------------------------------------------------------------------------##
##----------------------------End filling------------------------------##
##------------------------------------------------------------------------##
```

Visualize the loss plot for the train and validation datasets during training

- You are required to visualize the loss plots for the training and validation dataset during training.

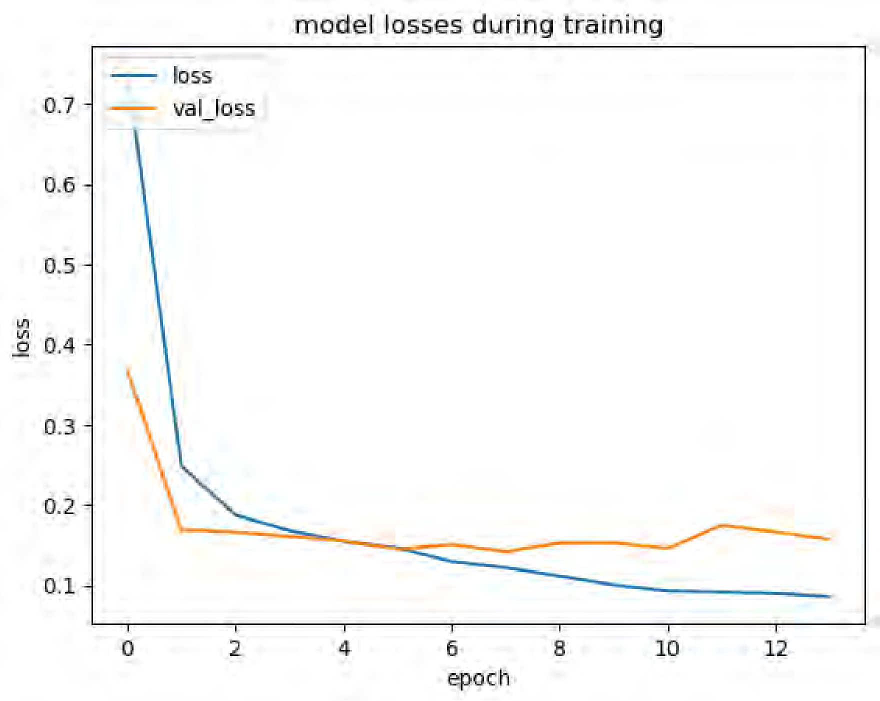
```
##------------------------------------------------------------------------##
##--------------------Students start filling below----------------------##
##------------------------------------------------------------------------##

"""
    Plot the loss of the training and validation datasets changes during the training.
    The information of the loss during the training is stored in "history" (return from model.fit()).
    Useful resource: https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/
"""


""" Filling code below """


""" End Filling """

##------------------------------------------------------------------------##
##-----------------------------End filling------------------------------##
##------------------------------------------------------------------------##
```

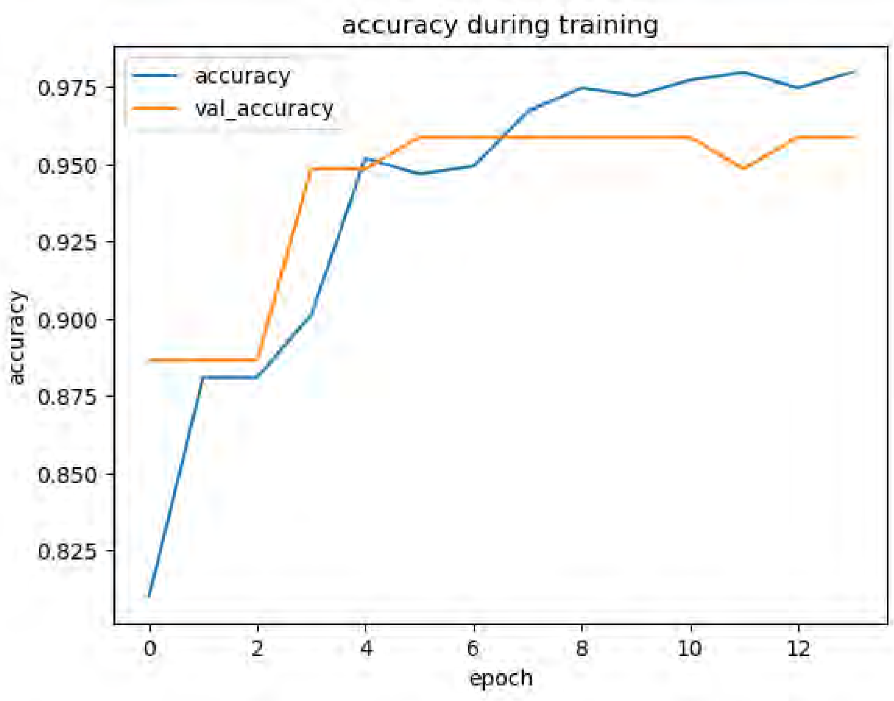
Visualize the accuracy plots for the train and validation datasets during training

- You are required to visualize the accuracy plots for the training and validation dataset during the train.

```
##------------------------------------------------------------------------##
##--------------------Students start filling below----------------------##
##------------------------------------------------------------------------##

"""
    Plot the accuracy on the training and validation datasets during the training.
    The information of the accuracy during the training is stored in "history" (return from model.fit()).
    Useful resource: https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/
"""


""" Filling code below """


""" End Filling """

##------------------------------------------------------------------------##
##-----------------------------End filling-------------------------------##
##------------------------------------------------------------------------##
```

Example Loss and accuracy visualization





- Useful tools: https://matplotlib.org/stable/tutorials/pyplot.html