Q1.

In the following case of working with sensor readings over time, a hybrid model consisting of both sequence-to-sequence and sequence-to-vector prediction approaches would be suitable to predict the operating mode of a wind turbine. The sequence-to-sequence prediction approach allows the model to learn from the sequence sensor readings over time. In the given dataset, the time_series_1 and time_series_2 arrays represent 5,000 sensor readings from a turbine over time by one of two sensors. By using sequence-to-sequence models, the model can learn from the temporal patterns in the sensor readings and potentially identify patterns that indicate a fault in the turbine.

Further, the sequence-to-vector approach would associate the sensor readings to a single output determining the operating mode of the wind turbine (indicated by the multi-class output of 0,1,2 and 3). The neural network models created also incorporate the same hybrid model wherein the initial RNN layers have *'return_sequence'* being set to TRUE (assuming a sequence-to-sequence approach), feeding the output (considering the entire sequence of time steps) to the last RNN layer (which considers only the immediately preceding time step) and thereafter predicts its association to one of the four classes.

Therefore, a hybrid model containing both approaches will be a suitable one.

The data shape (input) used for the approach defined above will be in the format *[batch size, number of time steps, number of features]* i.e., in the form of a 3D tensor. As given in the dataset –

- Batch Size = Number of turbine runs i.e., 4000
- Number of time steps = Number of observations per turbine run i.e., 5000
- Number of features = Features of the time series i.e., 2 (pitch angle, and generator torque)

Therefore, the data shape to be used for inputs is [4000,5000,2].

Q2.

Please refer to the Python notebook given.

Q3.

Model#	Model Architecture	Validation Set Results	
		Loss	Accuracy
1	**BASELINE** - Simple feed-forward network with 4 FCNN dense layers	0.45	0.78
2	2 GRU layers followed by 2 FCNN dense layers	0.4	0.82
3	3 Conv1D layers + 2 GRU layers + 1 FCNN dense layer	0.36	0.82
4	3 Conv1D layers + 2 LSTM layers + 1 FCNN dense layer	0.32	0.87

Note - Model# references can be found on the code as well

For Q3, the baseline model has been defined as a simple feed-forward dense neural network model with 4 dense layers. The other three neural networks have been created with a combination of CNN and RNN networks (Model #2 includes only RNN), along with fully connected neural networks at the end (using the Softmax activation function).

We use a combination of CNNs and RNNs to model the time-series data and classify the operating mode of the turbines because CNNs can extract features (more efficiently than fully connected neural networks) and RNNs can model temporal dependencies between these features, resulting in better classification accuracy. Also, LSTM and GRU RNNs have been used to overcome the vanishing gradient problem, which is important in our case since sensor readings are over time, and longer-term dependencies can be recognised.

Note – Simple RNN was another possible neural network that could have been used but has not been explored due to extremely high computing time.

Also, while comparing the models used in Q3, Models #2, #3 and #4 have been able to gather a higher accuracy and a lower loss value for the validation dataset in comparison to the baseline model.

Q4.

Our neural network models mentioned in the previous question include the application of both Convolution 1D as well as RNN layers, and as per our observation, they have a very high accuracy in predicting the operating mode of the wind turbines through using sensor signals, which further affirms the fact that both are useful in case of dealing with time series data. Though some fundamental differences exist, and they are as follows –

- Convolutional 1D, which is more computationally efficient than its 2D and 3D counterparts due to lower spatial dimensions, is used to identify patterns or trends within sections of time series data and predict based on those patterns.
- On the other hand, RNNs are most useful in identifying dependencies (CNN does not identify dependencies between data points) between time series data points across previous time steps (observations) and are used to predict the state in the succeeding time steps. Also, since RNNs identify dependencies across past time steps, storing the memory across time steps is equally important to determine the state in succeeding time steps (any lack of identifying long-term dependencies is resolved through the usage of LSTM and GRU).

We use Convolutional 1D in our case to look at patterns of sensor signals and feed the output to RNNs so that they can determine dependencies between past time steps and predict the state in succeeding time steps.

Business applications of the outputs generated by Convolutional 1D neural networks for our time series data of sensor signals include -

- Weather Analysis (image conversion of our time series data can help analyse wind patterns and predict how will they affect wind turbine performance)
- Supply chain management (any patterns of emergencies will help planning to avoid any risks and delays)
- Optimizing power generation (real-time adjustment of wind turbine pitch angles can help maximise wind power generation)

Business applications of the outputs generated by RNN for our time series data of sensor signals include -

- Forecasting wind energy production (based on current weather conditions/wind speed)
- Energy Resource Planning (high winds may increase downtime, so preservation may be necessary)
- Customer service readiness (periods of extreme weather conditions involve a higher number of customer contacts, requiring preparedness in terms of customer service)

Q5 –

Time-series data poses a significant challenge due to the presence of signal noise, resulting in high volatility. To overcome this issue, CNN can classify even the slightest anomalies and signal noise in benchmark data. In wind turbines, CNN can handle the complex dynamics and environmental disturbances in both onshore and offshore machines. The time-series data structure can be transformed into 2D images using CNN, and the information can be extracted by analysing the relationship between adjacent pixels in the image matrix.

Additionally, CNN is more computationally efficient than RNN as it can learn patterns in batches, which enables parallelized training. RNN, on the other hand, trains data sequentially, requiring the completion of previous computations before moving on to the next. Another advantage of CNN is that it considers the distribution of the data points, unlike RNN, which looks at information before the timestep of prediction.

Q6 –

Please refer to the Python notebook given

Q7 –

Model#	**Model Architecture**	**Validation Set Results**	
		Loss	**Accuracy**
5	3 Conv 2D layers + 1 Dropout layer + 1 Max Pooling layer + 1 FCNN layer (*Fig 12 of paper*)	0.23	0.9

Note - Model# references can be found on the code as well

The model given in Figure 12 of the paper was replicated for our time series data, giving us an accuracy of 0.9 and a loss of 0.23. In comparison to the baseline model, this model is a better fit for our time series dataset. Please refer to the Python notebook for more information.

Q8 –

For the following question, we have used two pre-trained neural network models, namely *MobileNet* and *Inception V3.* The purpose of using these two pre-trained models is due to their enhanced ability to process image classification (as mentioned in the paper on image-based CNN analysis, as well as done in the previous question).

The results were as follows –

Model#	**Model Architecture**	**Validation Set Results**	
		Loss	**Accuracy**
6	Pretrained *MobileNet* architecture	0.34	0.86
7	Pretrained *InceptionV3* architecture	0.51	0.82

Note - Model# references can be found on the code as well

Both the pre-trained neural network models have a better for our dataset in comparison to the baseline. Please refer to the Python notebook for further information.

Q9 –

Model#	Model Architecture	Validation Set Results	
		Loss	Accuracy
1	**BASELINE** - Simple feed-forward network with 4 dense layers	0.45	0.78
2	2 GRU layers followed by 2 dense layers	0.4	0.82
3	3 Conv1D layers + 2 GRU layers + 1 dense layer	0.36	0.82
4	3 Conv1D layers + 2 LSTM layers + 1 dense layer	0.32	0.87
5	3 Conv 2D layers + 1 Dropout layer + 1 Max Pooling layer + 1 FCNN layer (*Fig 12 of paper*)	0.23	0.9
6	Pretrained *MobileNet* architecture	0.34	0.86
7	Pretrained *InceptionV3* architecture	0.51	0.82

Note - Model# references can be found on the code as well

The neural network model highlighted in red is the best model due to the following reasons –

- The highest validation accuracy of 0.9 ensures higher chances of correct classification of the sensor signals to one of the four classes of wind turbine health.
- The lowest validation loss of 0.23 ensures minimal overfitting.

Therefore, the following model is trained on the combined training and validation dataset. The accuracy score and loss on the test dataset is 0.895 and 0.25, respectively.