Project Week 1

22.05.2024

Garance Boesinger, Daniel Elmaleh, Nathalie Künzle

Given architecture

- I have a dataset consisting of camera images from two different locations of a campus over time and the labels of these images are the GHI values on the campus 2 hours after the images are taken. Provide a keras neural network for using these images to predict the labels of these images, don't forget to utilize the time aspect of the data by using some RNN layers.

```
# Create a Sequential model
model = models.Sequential()
# Use TimeDistributed to apply CNN layers on each time step
model.add(layers.TimeDistributed(layers.Conv2D(32, (3, 3), activation='relu'),
input_shape=(time_steps, *input_shape)))
model.add(layers.TimeDistributed(layers.MaxPooling2D((2, 2))))
model.add(layers.TimeDistributed(layers.Conv2D(64, (3, 3), activation='relu')))
model.add(layers.TimeDistributed(layers.MaxPooling2D((2, 2))))
model.add(layers.TimeDistributed(layers.Conv2D(128, (3, 3), activation='relu')))
model.add(layers.TimeDistributed(layers.MaxPooling2D((2, 2))))
# Flatten and Dense layers for each time step
model.add(layers.TimeDistributed(layers.Flatten()))
model.add(layers.TimeDistributed(layers.Dense(dense_units, activation='relu')))
# LSTM layers to capture temporal dependencies
model.add(layers.LSTM(lstm_units, return_sequences=True))
model.add(layers.LSTM(lstm_units)) # Final Dense layer to output the GHI value
model.add(layers.Dense(1))
 # Predicting a single scalar value for each sequence
 # Compile the model
model.compile(optimizer='adam', loss='mse', metrics=['mae']) return model
```

Overview: Tested input variations

For each test, we used normalized images, normalized labels and normalized meteo data.

A) only one camera as input

- Approach 1: One frame only
- Approach 2: 2 frames, same camera

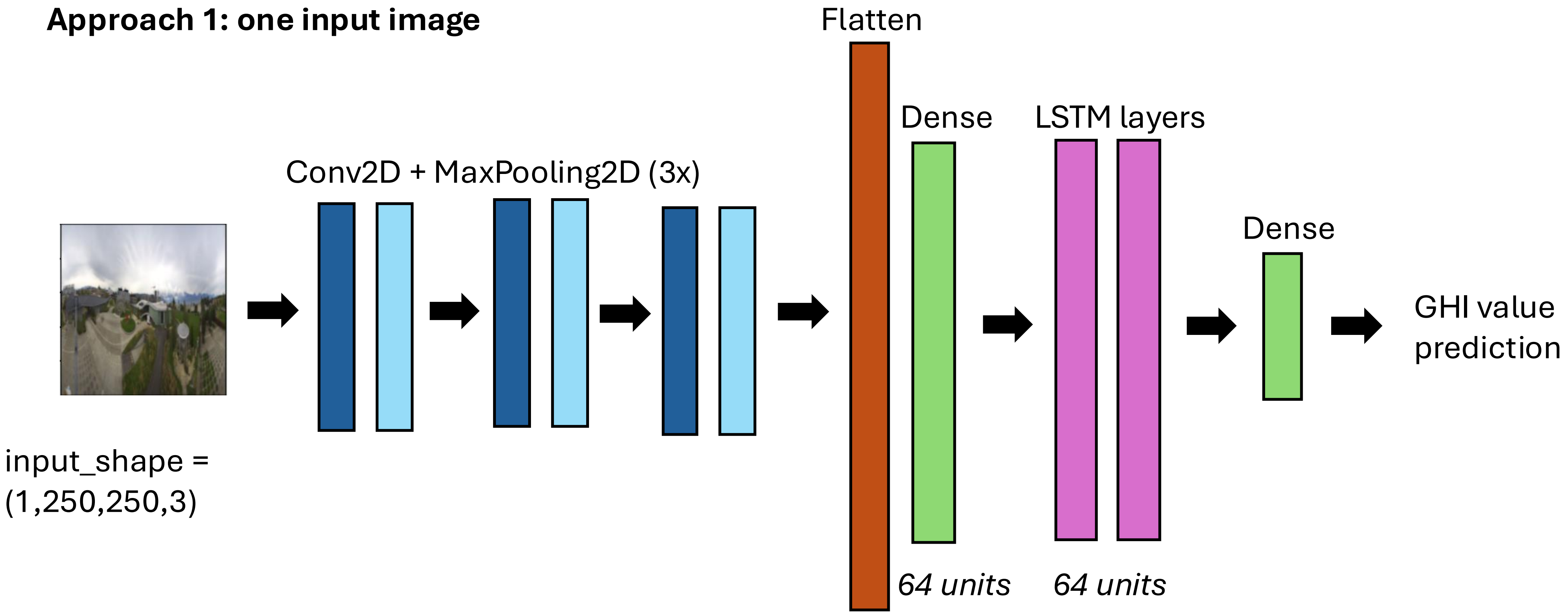
B) 2 frames : one image per camera as input

C) (one image per camera)* wind speed coefficients

D) one image per camera + meteo params (wind speed + cloud opacity)

- Approach 1: Meteo parameters as a second input in vectors
- Approach 2: Meteo params as another channel in the input (broadcasted input dataset)

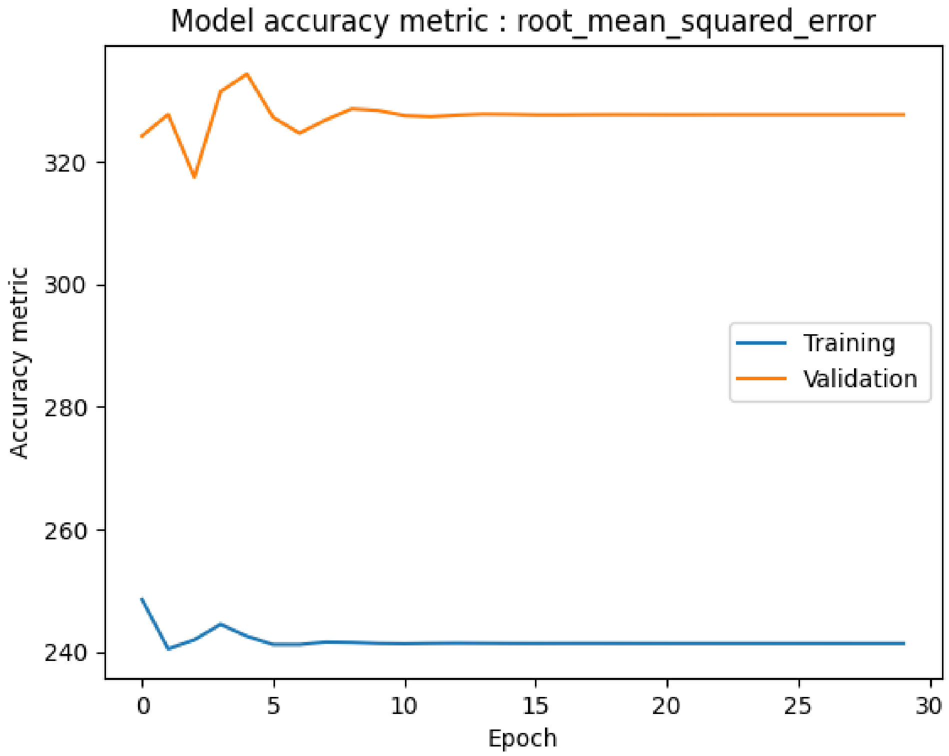
A) Only one camera as input



A) Only one camera as input

Approach 1: one input image

input_shape = (2,250,250,3)




```
* Evaluating the performance of the trained network on the unseen test dataset *
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.1710 - root_mean_squared_error: 0.5761
Error - root_mean_squared_error: 272.978
```

Parameters:

- Nb of train samples: 800
- Nb of validation samples: 300
- Nb of test samples : 800

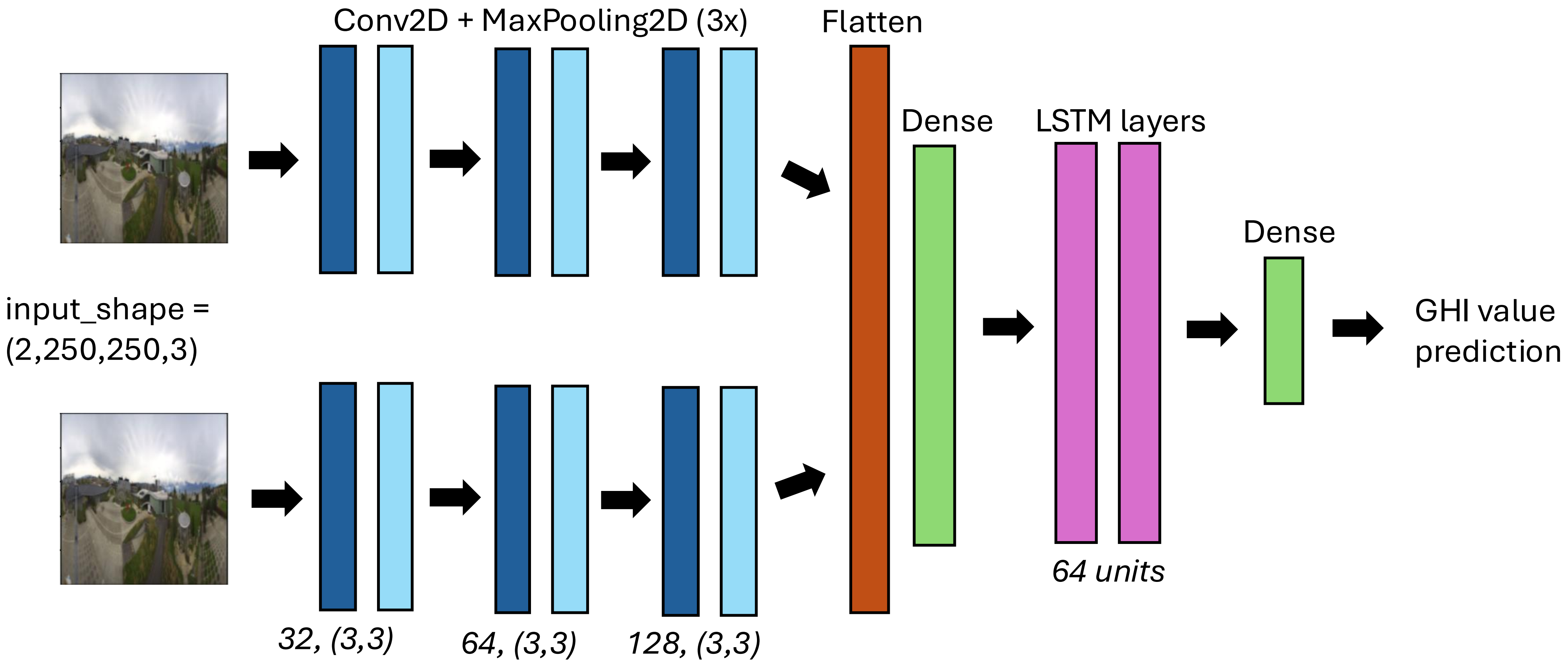
- input_shape = (2,250,250,3)
- batch_size = 100
- max_epochs = 30

- Activation function: Relu
- Loss: MSE
- Optimizer: Adam

Test RMSE = 273

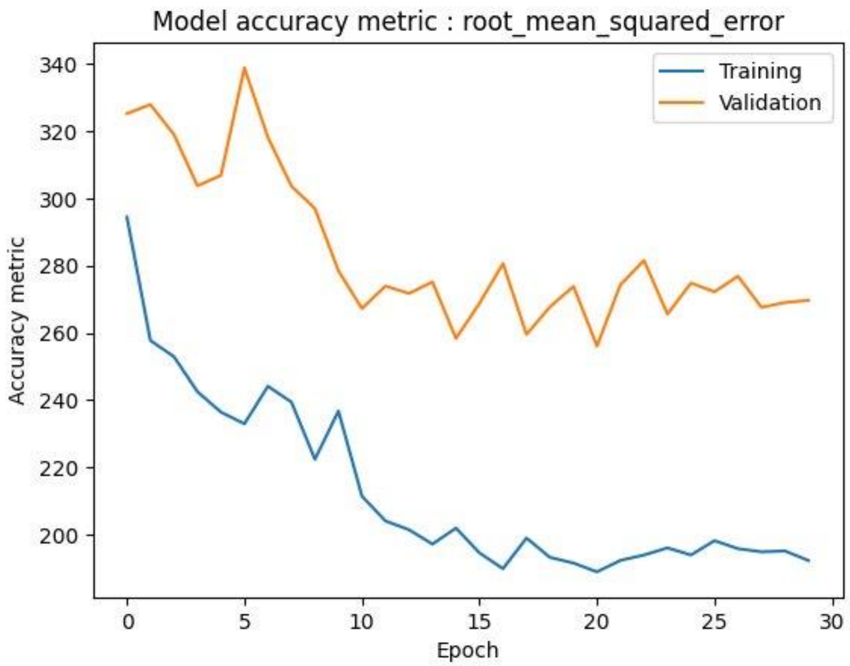
A) Only one camera as input

Approach 2: two times the same image as input



A) Only one camera as input

Approach 2: two times the same image as input
input_shape = (2,250,250,3)




```
* Evaluating the performance of the trained network on the unseen test dataset *
10/10 ━━━━━━━━━━━━━━━━━━━━ 4s 122ms/step - loss: 0.0876 - root_mean_squared_error: 0.3930
Error - root_mean_squared_error: 230.753
```

Parameters:

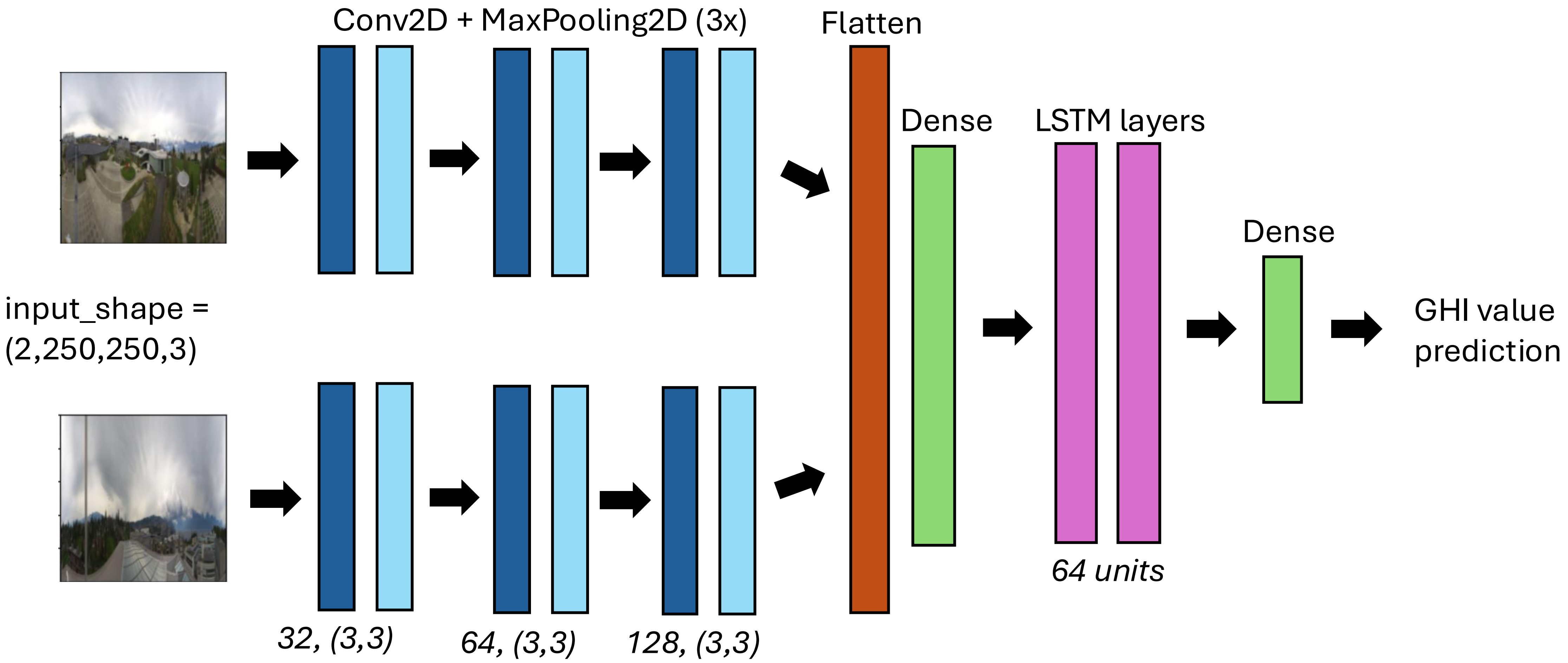
- Nb of train samples: 600
- Nb of validation samples: 300
- Nb of test samples: 300

- input_shape = (2,250,250,3)
- batch_size = 100
- max_epochs = 30

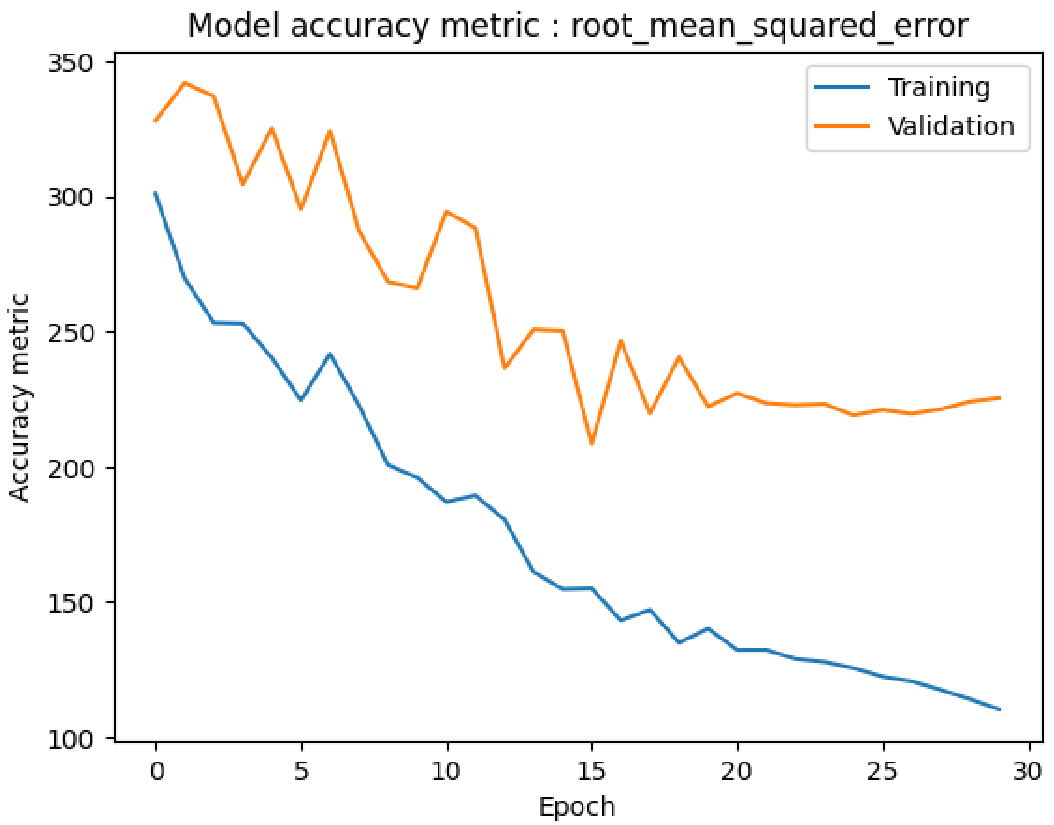
- Activation function: Relu
- Loss: MSE

Test RMSE = 230

B) 2 frames : one image per camera as input



B) 2 frames : one image per camera as input




```
* Evaluating the performance of the trained network on the unseen test dataset *
10/10 ━━━━━━━━━━━━━━━━━━━━ 4s 118ms/step - loss: 0.0144 - root_mean_squared_error: 0.1621
Error - root_mean_squared_error: 84.667
```

Parameters:

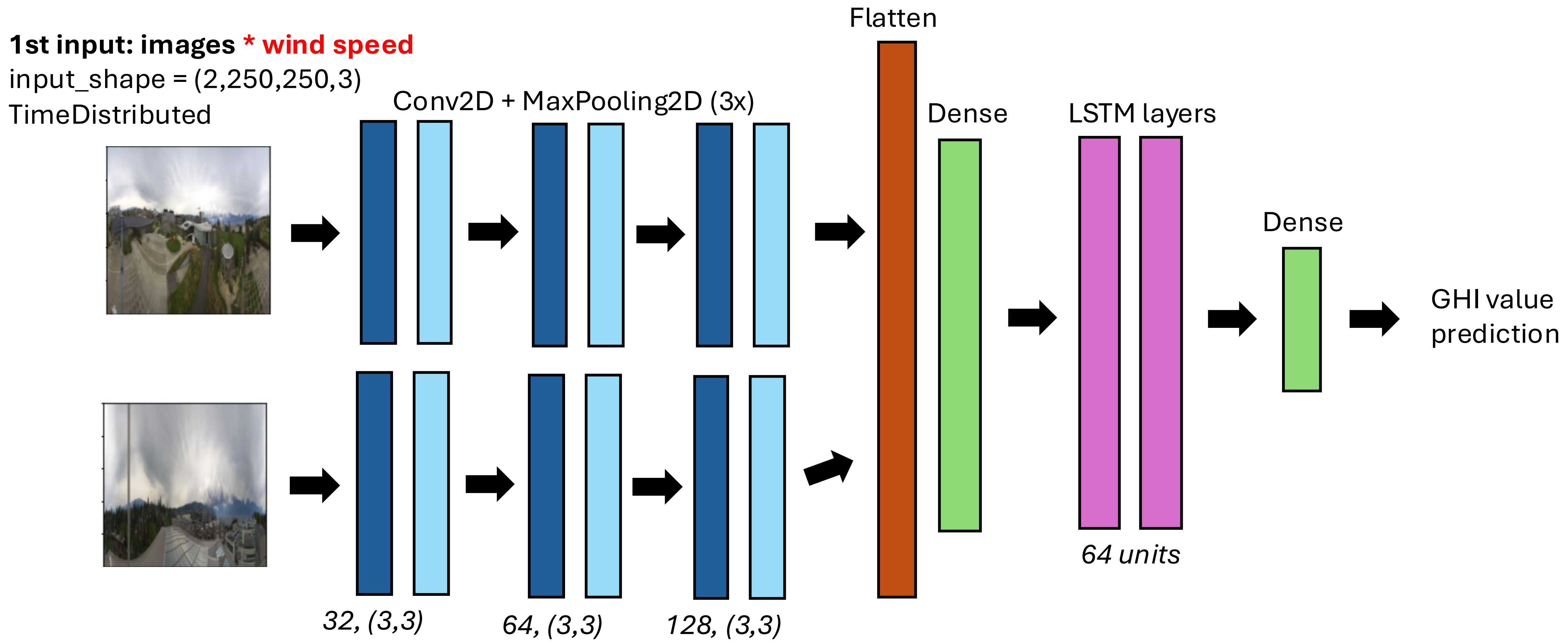
- Nb of train samples: 600
- Nb of validation samples: 300
- Nb of test samples: 300

- input_shape = (2,250,250,3)
- batch_size = 100
- max_epochs = 30

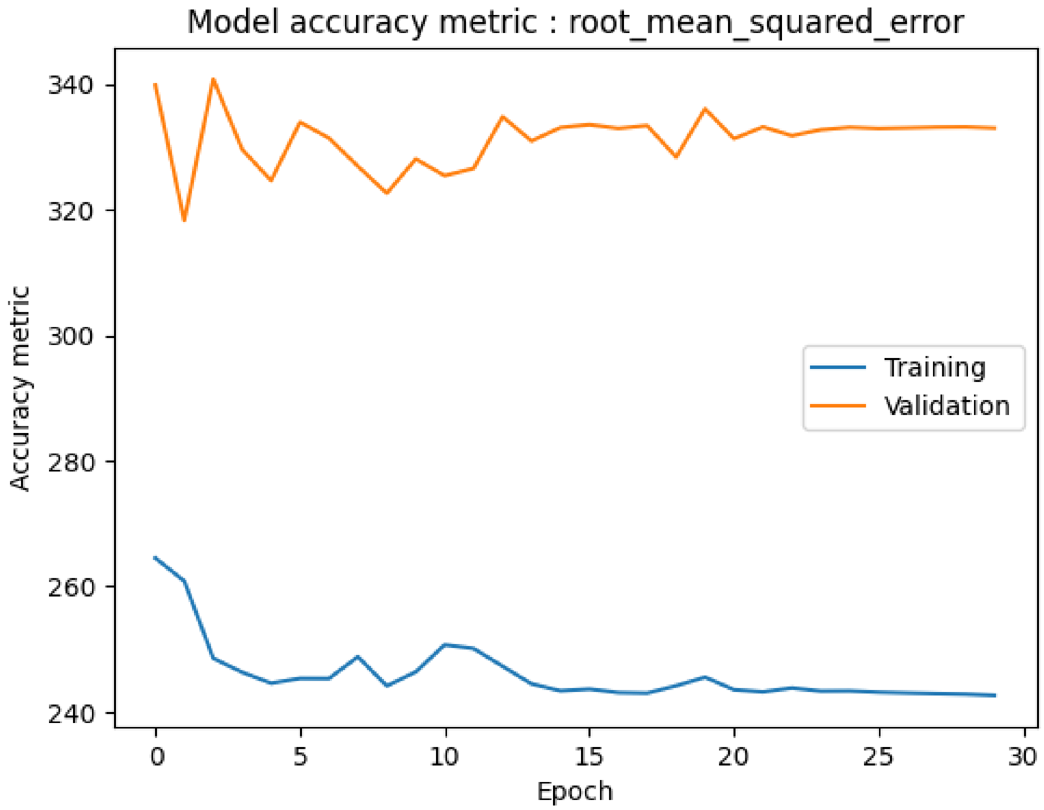
- Activation function: Relu
- Loss: MSE
- Optimizer: Adam

Test RMSE = 85

C) (one image per camera) * wind speed as coef



C) (one image per camera) * wind speed as coef




```
* Evaluating the performance of the trained network on the unseen test dataset *
25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 80ms/step - loss: 0.1715 - root_mean_squared_error: 0.5774
Error - root_mean_squared_error: 276.807
```

Parameters:

- Nb of train samples: 800
- Nb of validation samples: 300
- Nb of test samples: 800

- input_shape = (2,250,250,3)
- batch_size = 60
- max_epochs = 30

- Activation function: Relu
- Loss: MSE

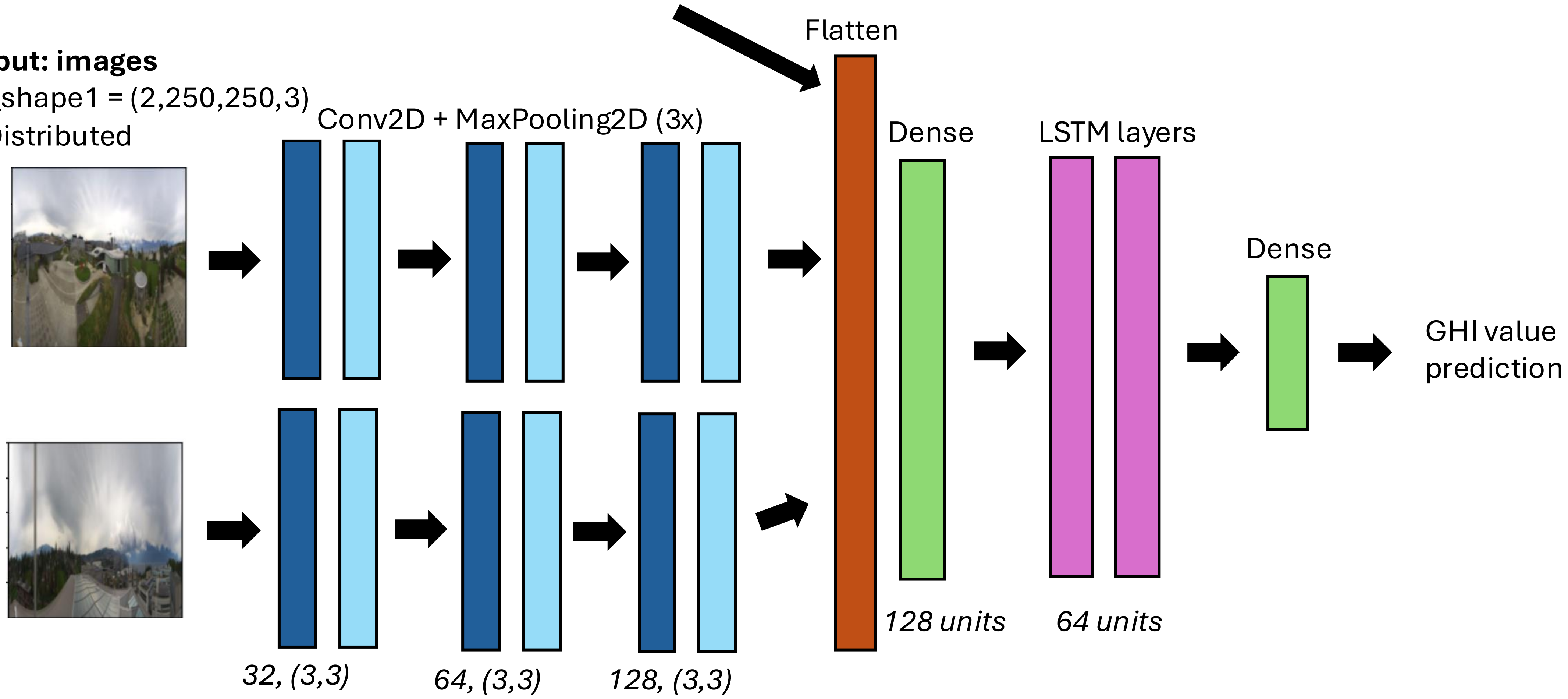
Test RMSE = 277

D) one image per camera + meteo parameters

Approach 1:

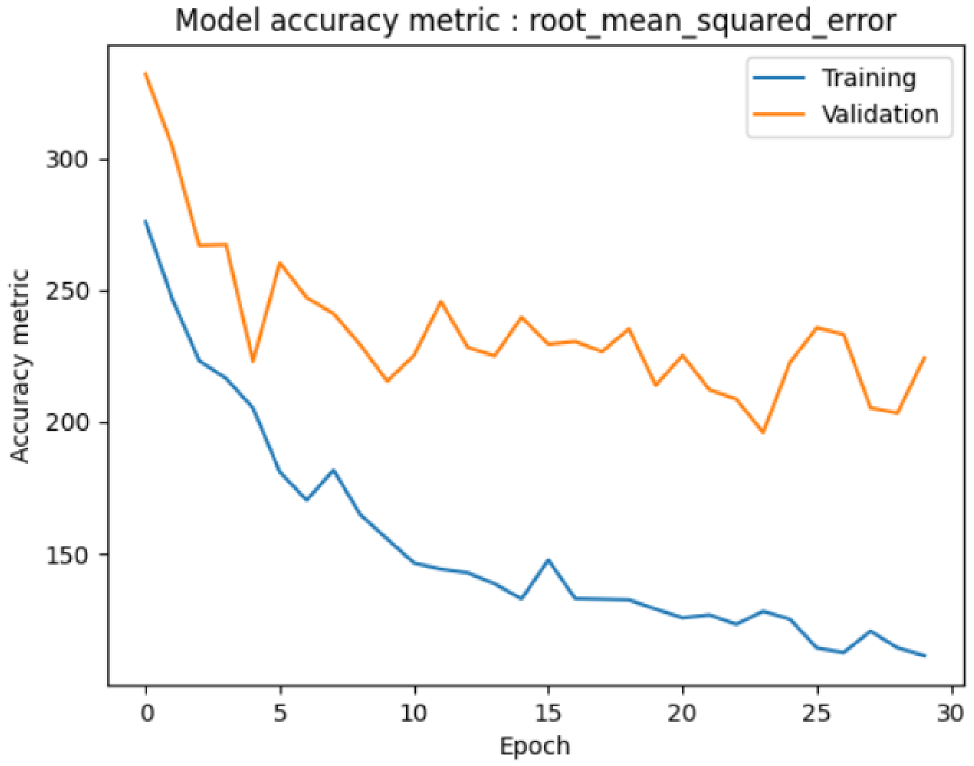
2nd input: meteo data
input_shape2 = (2,8)
For each image: 4x wind speed + 4x cloud opacity

1st input: images
input_shape1 = (2,250,250,3)
TimeDistributed



D) one image per camera + meteo parameters

Approach 1:




```
* Evaluating the performance of the trained network on the unseen test dataset *

25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 74ms/step - loss: 0.0382 - root_mean_squared_error: 0.2676

Error - root_mean_squared_error: 169.334
```

Parameters:

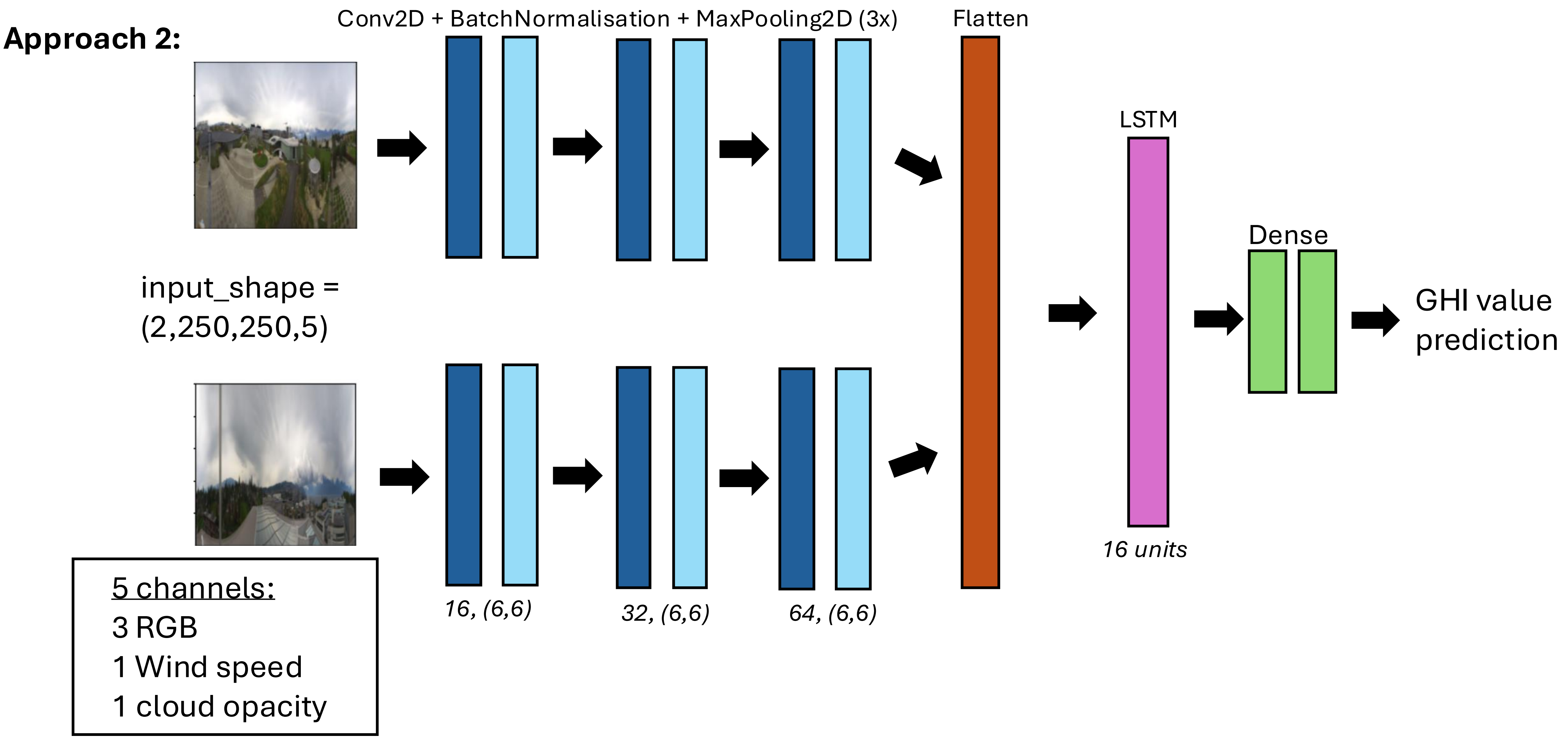
- Nb of train samples: 800
- Nb of validation samples: 300

- input_shape1 = (2,250,250,3)
- Input_shape2 = (2,8)
- batch_size = 100
- max_epochs = 30

- Activation function: Relu
- Loss: MSE

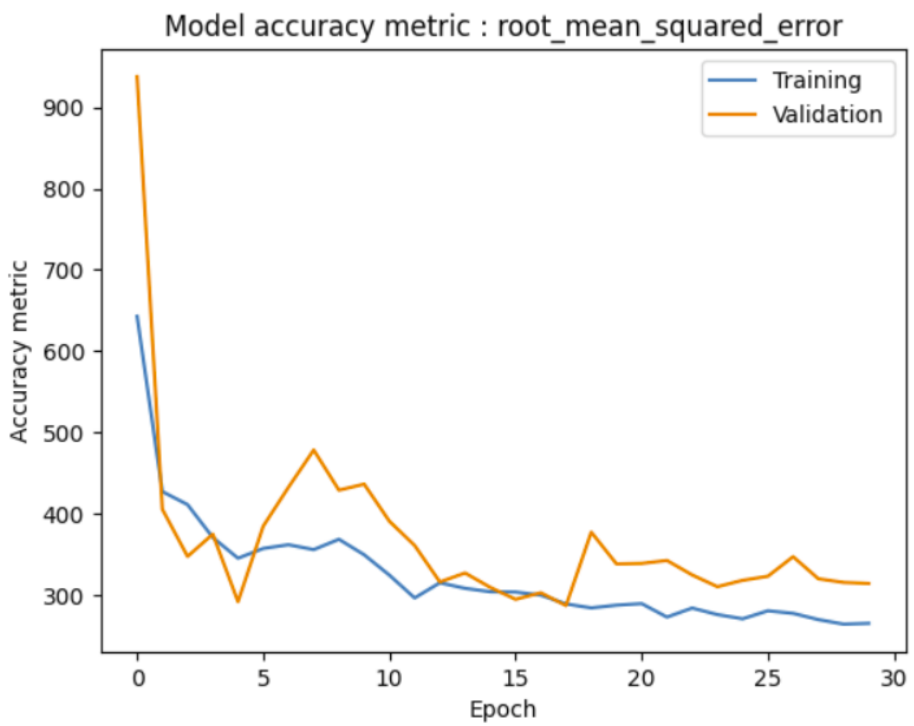
Test RMSE = 169

D) one image per camera + meteo parameters



D) one image per camera + meteo parameters

Approach 2:




```
* Evaluating the performance of the trained network on the unseen test dataset *

7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.2844 - root_mean_squared_error: 0.7230

Error - root_mean_squared_error: 375.524
```

Parameters:

- Nb of train samples: 500
- Nb of validation samples: 200
- Nb of test samples: 200

- input_shape1 = (2,250,250,5)
- batch_size = 100
- max_epochs = 30

- Activation function: LeakyRelu
- Loss: MSE
- Optimizer: Adam

Test RMSE = 376

Overview: Tested input variations

A) only one camera as input

- Approach 1: One frame only – RMSE: 273
- Approach 2: 2 frames, same camera – RMSE: 230

B) 2 frames : one image per camera as input – RMSE: 85

Conclusion 1 : Real improvement when using **2 frames** and using **both cameras**

C) (one image per camera)* wind speed coefficients – RMSE: 277

D) one image per camera + meteo parameters (wind speed + cloud opacity)

- Approach 1: Meteo parameters as a second input in vectors – RMSE: 169
- Approach 2: Meteo parameters as another channel in the input (broadcasted input dataset) – RMSE: 376

Conclusion 2 : (Relative) improvement when using **meteo data**, but very sensitive to the way we incorporated it into the network.

Project Week 2

22.05.2024

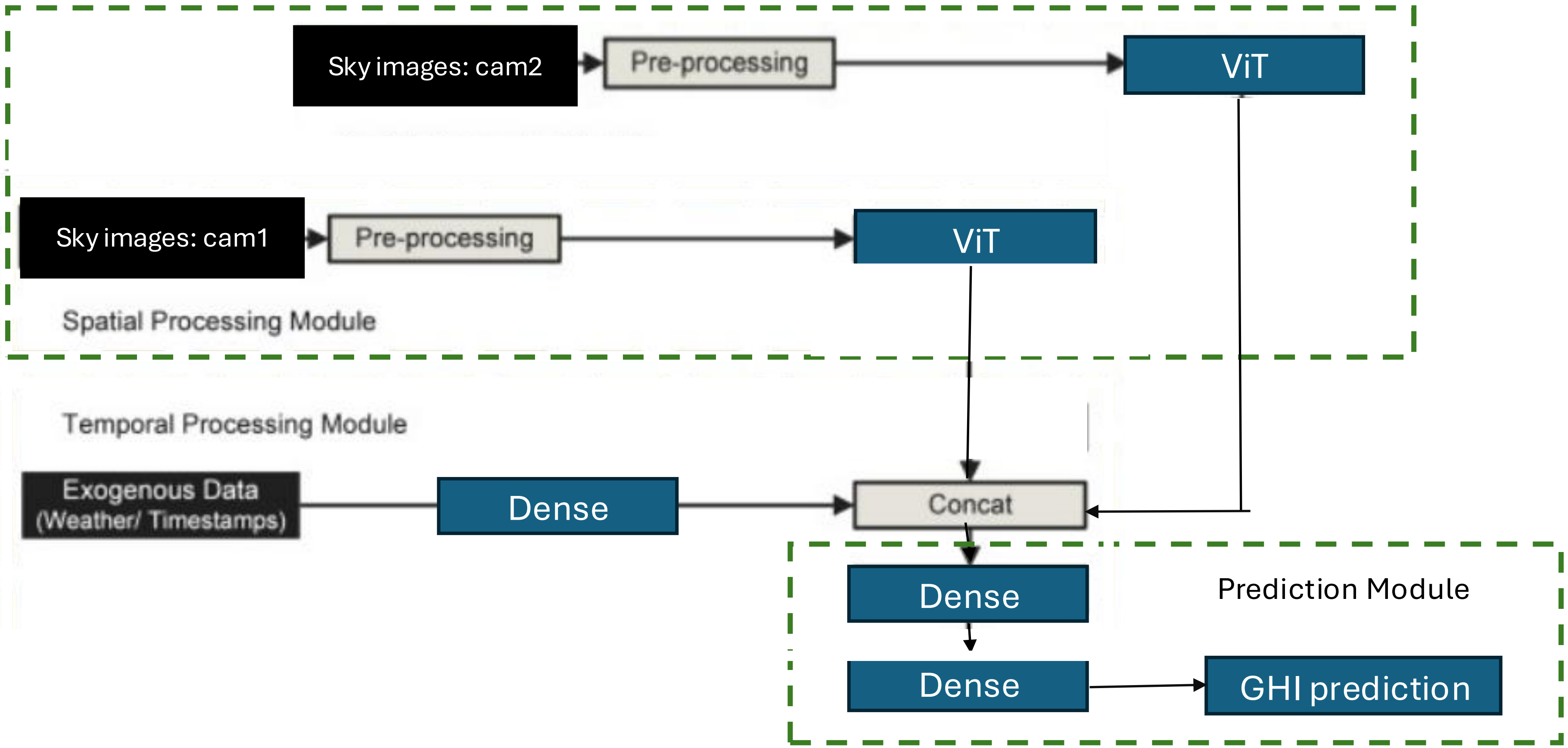
Group 17

Garance Boesinger, Daniel Elmaleh, Nathalie Künzle

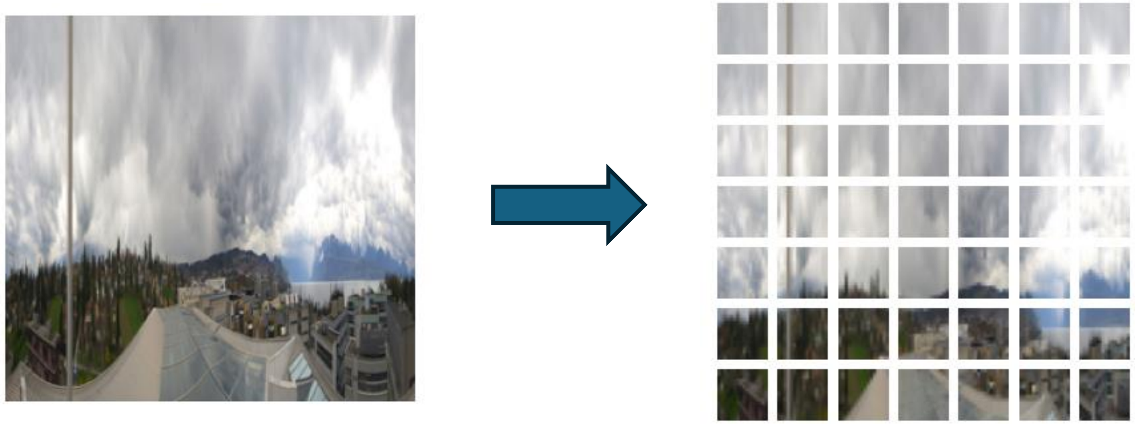
Model Overview

- **Purpose**: Combining Vision Transformer (ViT) architecture with traditional neural network layers, to predict GHI.
- **Input Features**:
  - Images from two sources (labeled as M and BC).
  - monthly and hourly data.
  - meteorological data.
- Data was reshaped and standardize to be homogeneous.

Model architecture



Model Architecture



- **Data Augmentation**
- **Patch Creation**
- **Patch Encoding**
- **Transformer Blocks**:
  - Multi-head self-attention & multi-layer perceptrons (MLP).
  - Utilizes skip connections and layer normalization for stability and efficiency.
- **Feature Extraction**:
  - Outputs from transformer blocks are flattened and passed through dropout layers to reduce overfitting.
  - Uses MLP for additional feature processing and dimensionality reduction.

Integration of Vision Transformer with Other Inputs

- **Separate ViT Models for Different Image Sources.**
- **Feature Concatenation**:
  - Both ViT paths.
  - Monthly/hourly data and meteorological data.
- **Final Feature Fusion**:
  - All features are concatenated into a unified feature vector.
  - Multiple dense layers with ReLU activation.
  - Output Layer.
- **Prediction of GHI**: Sigmoid activation.

Model Compilation

- **Optimizer**: Adam optimizer with a configurable learning rate.
- **Loss Function**: Mean Squared Error (MSE).
- **Metrics**: Root Mean Squared Error (RMSE).

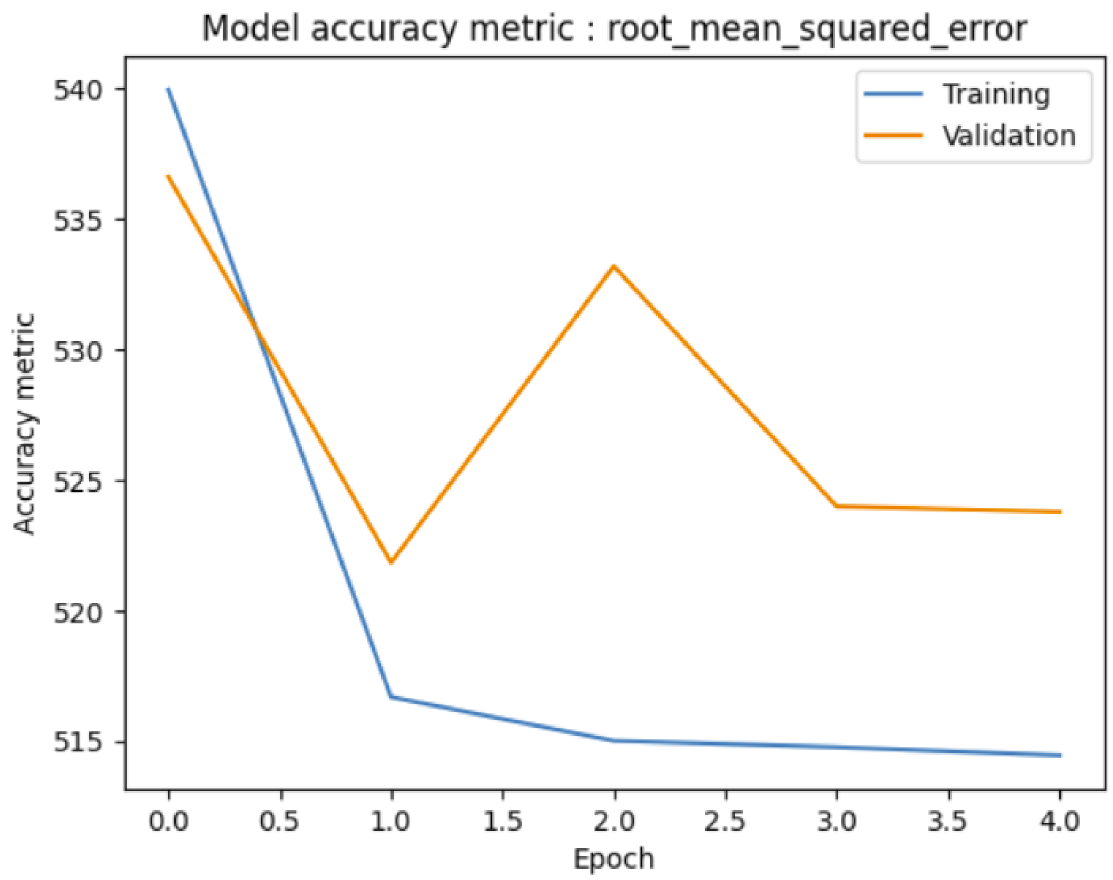
Hyperparameters

```python
lr = 0.002
loss = 'mse'
metric = [RootMeanSquaredError()]
batch_size = 32
max_epochs = 5


#ViT
weight_decay = 0.0001
image_size = 250  # We'll resize input images to this size
patch_size = 16
num_patches = (image_size // patch_size) ** 2
projection_dim = 64
num_heads = 8
transformer_units = [
    projection_dim * 2,
    projection_dim,
]  # Size of the transformer layers
transformer_layers = 8
mlp_head_units = [
    2048,
    1024,
]  # Size of the dense layers of the final classifier
```

Results and variants

1) All data

--> Concatenation after the flat layer



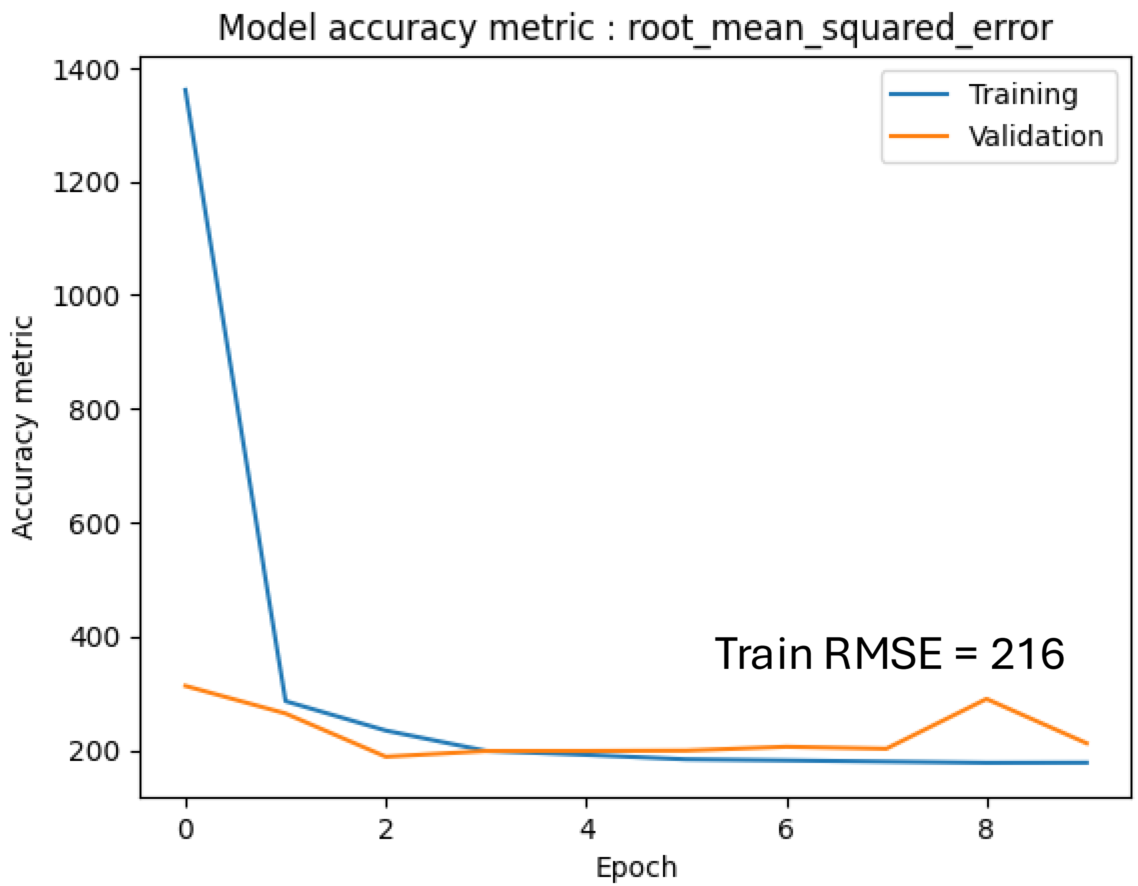

```
* Evaluating the performance of the trained network on the unseen test dataset *

93/93 ━━━━━━━━━━━━━━━━━━━━ 23s 245ms/step - loss: 0.1657 - root_mean_squared_error: 0.5677

Error - root_mean_squared_error: 271.673
Loss - mean_squared_error: 0.045
271.6731701493263
```

RMSE = 583

- images_shape = (250, 250, 3) (x2)
- lr = 0.002
- loss = 'mse'
- metric = [RootMeanSquaredError()]
- batch_size = 132
- max_epochs = 5

- weight_decay = 0.0001
- image_size = 250 # resized input images
- patch_size = 16
- num_patches = (image_size // patch_size) ** 2
- projection_dim = 64
- num_heads = 4

2.a) One frame, one camera as input (no meteo data)




```
* Evaluating the performance of the trained network on the unseen test dataset *

93/93 ━━━━━━━━━━━━━━━━━━━━ 11s 117ms/step - loss: 0.0538 - root_mean_squared_error: 0.3164

Error - root_mean_squared_error: 226.615
Loss - mean_squared_error: 0.049
[27]: 226.6152500808239
```

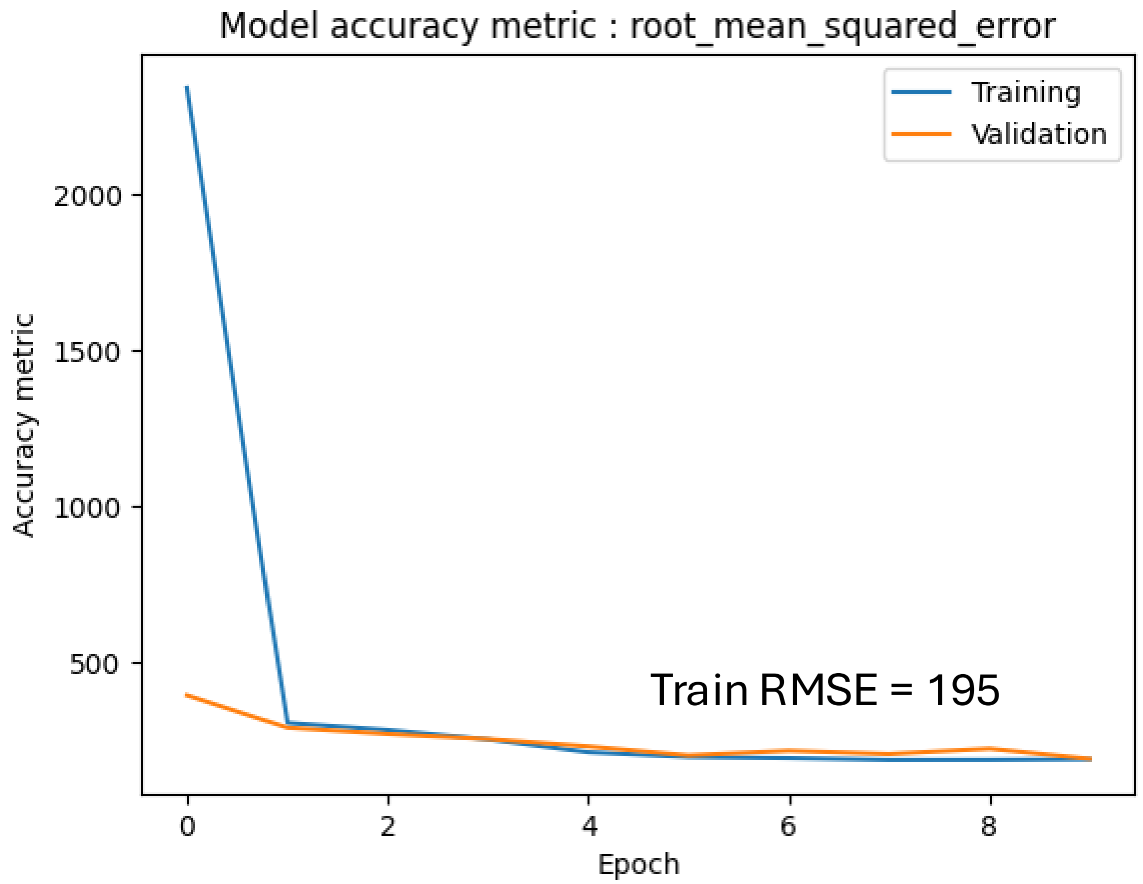
RMSE = 325

- images_shape = (250, 250, 3)
- lr = 0.002
- loss = 'mse'
- metric = [RootMeanSquaredError()]
- batch_size = 32
- max_epochs = 10

- weight_decay = 0.0001
- image_size = 125 # resized input images
- patch_size = 16
- num_patches = (image_size // patch_size) ** 2
- projection_dim = 64
- num_heads = 4

2.b) Two frames (one per camera) as input (no meteo data)

--> Concatenation after the flat layer




```
* Evaluating the performance of the trained network on the unseen test dataset *

93/93 ━━━━━━━━━━━━━━━━━━━━ 17s 186ms/step - loss: 0.0311 - root_mean_squared_error: 0.2447

Error - root_mean_squared_error: 141.781
Loss - mean_squared_error: 0.019
[32]: 141.78056870400906
```

RMSE = 256

- images_shape = (250, 250, 3) (x2)
- lr = 0.01
- loss = 'mse'
- metric = [RootMeanSquaredError()]
- batch_size = 100
- max_epochs = 10

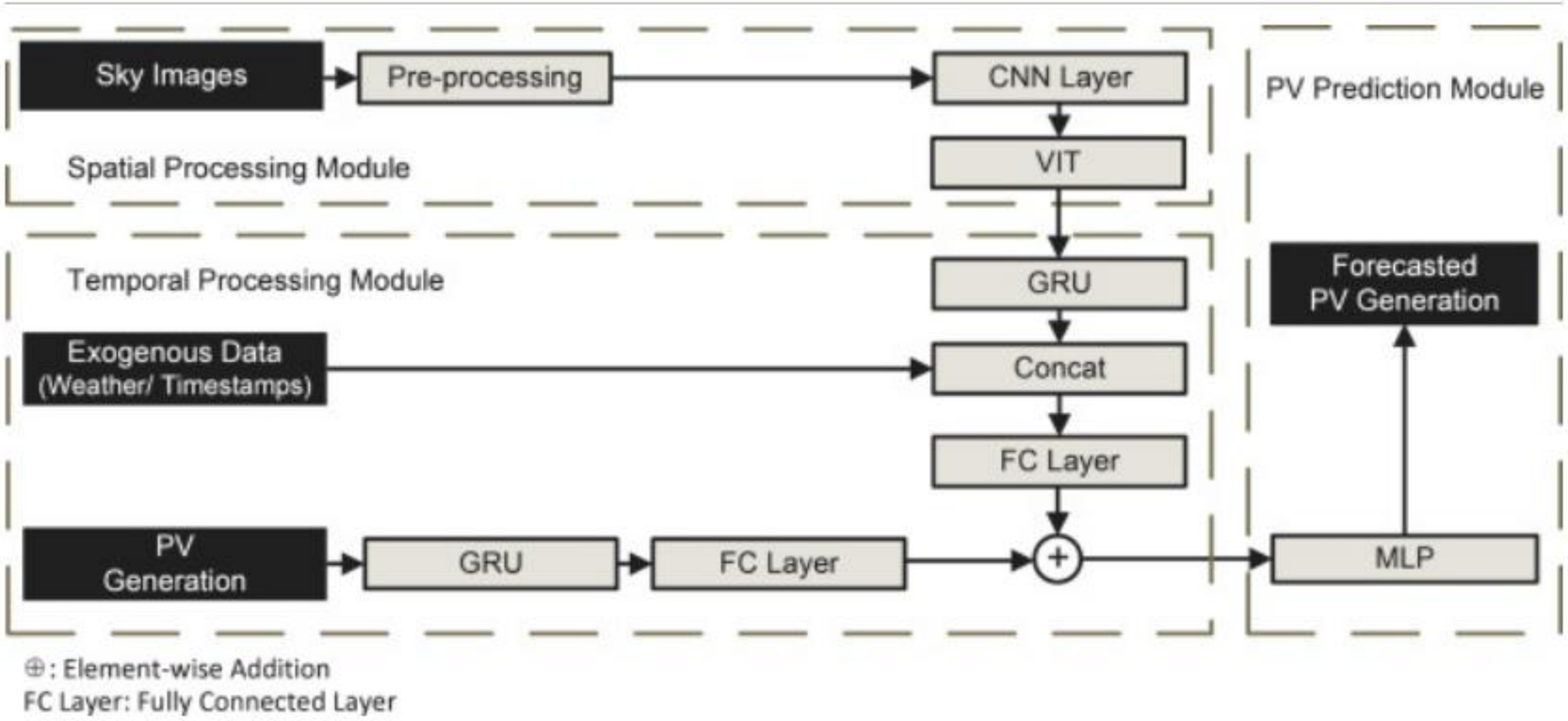
- weight_decay = 0.0001
- image_size = 125 # resized input images
- patch_size = 16
- num_patches = (image_size // patch_size) ** 2
- projection_dim = 64
- num_heads = 4

Attempts for a more complex model

- More layers
- **LSTM** for monthly and hourly data, meteorological data.
- Hyperparameters tuning

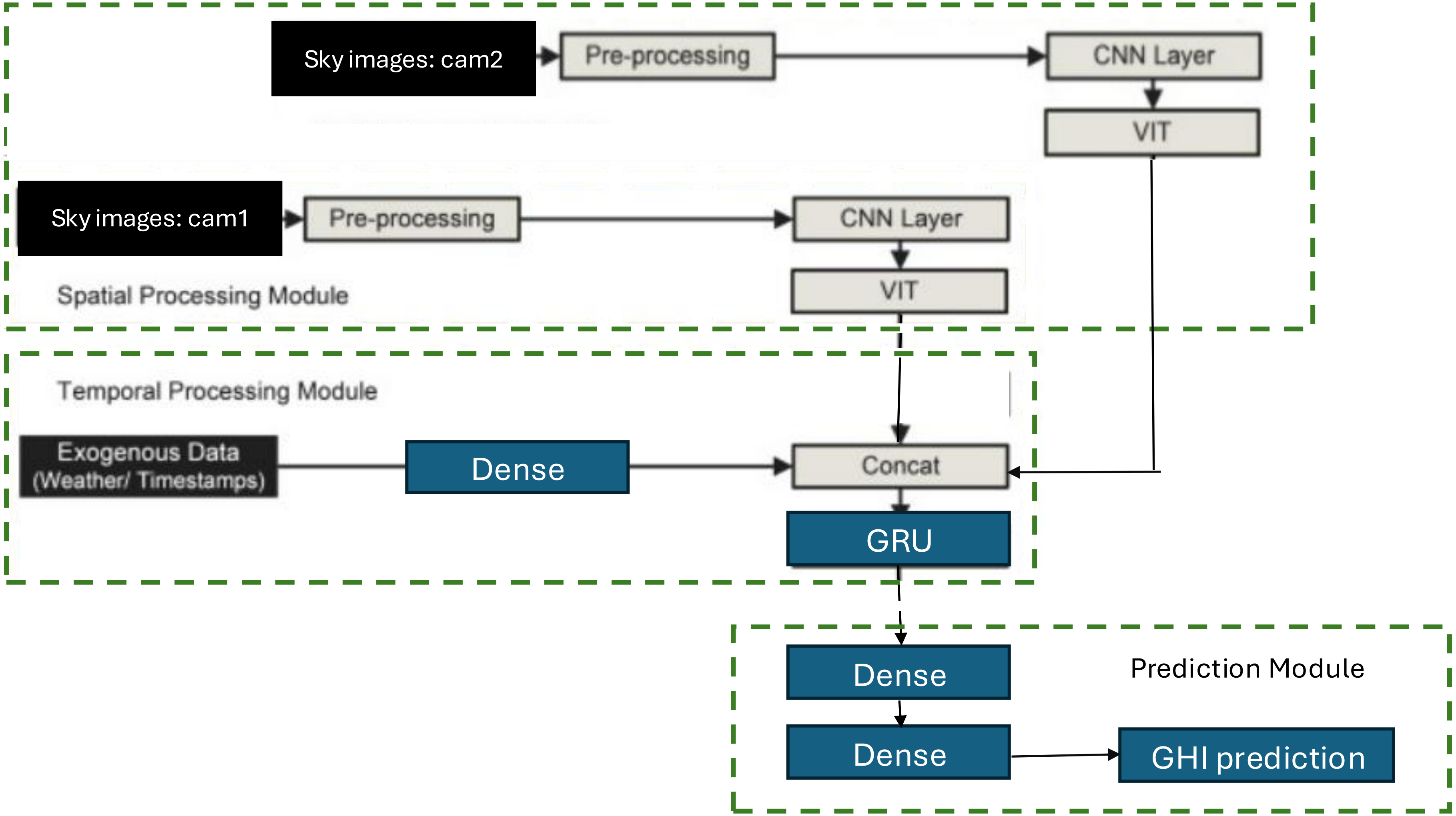
- Results: Worse

Inspiration from another model :

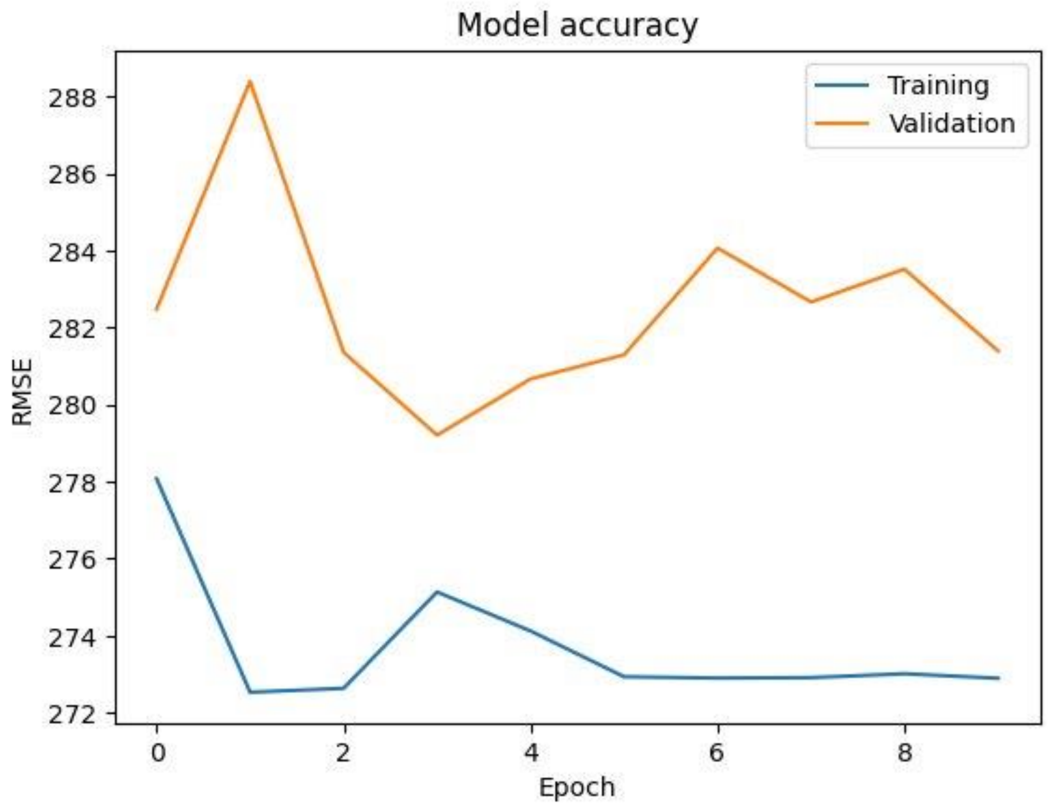


Source: Shijie Xu et al, "On vision transformer for ultra-short-term forecasting of photovoltaic generation using sky images," in Solar Energy, Vol 7, 2024, https://doi.org/10.1016/j.solener.2023.112203.

Our implementation :



Performance




```
* Evaluating the performance of the trained network on the unseen test dataset *

93/93 ━━━━━━━━━━━━━━━━━━━━ 21s 222ms/step - loss: 0.1355 - mse: 0.1355

Error - root_mean_squared_error: 249.801
```

RMSE: 378

4. Recap

Model	RMSE Train/Test score
1. Simple ViT, two frames + meteo without optimization	515 / 583 (model not optimized)
2.1. Simple ViT, One frame	216 / 325
2.2. Simple ViT, two frames	195 / 256
3. Two parallel ViTs for images + meteo data through GRU layer	273 / 378 (model not optimized)

Project Week 3

29.05.2024

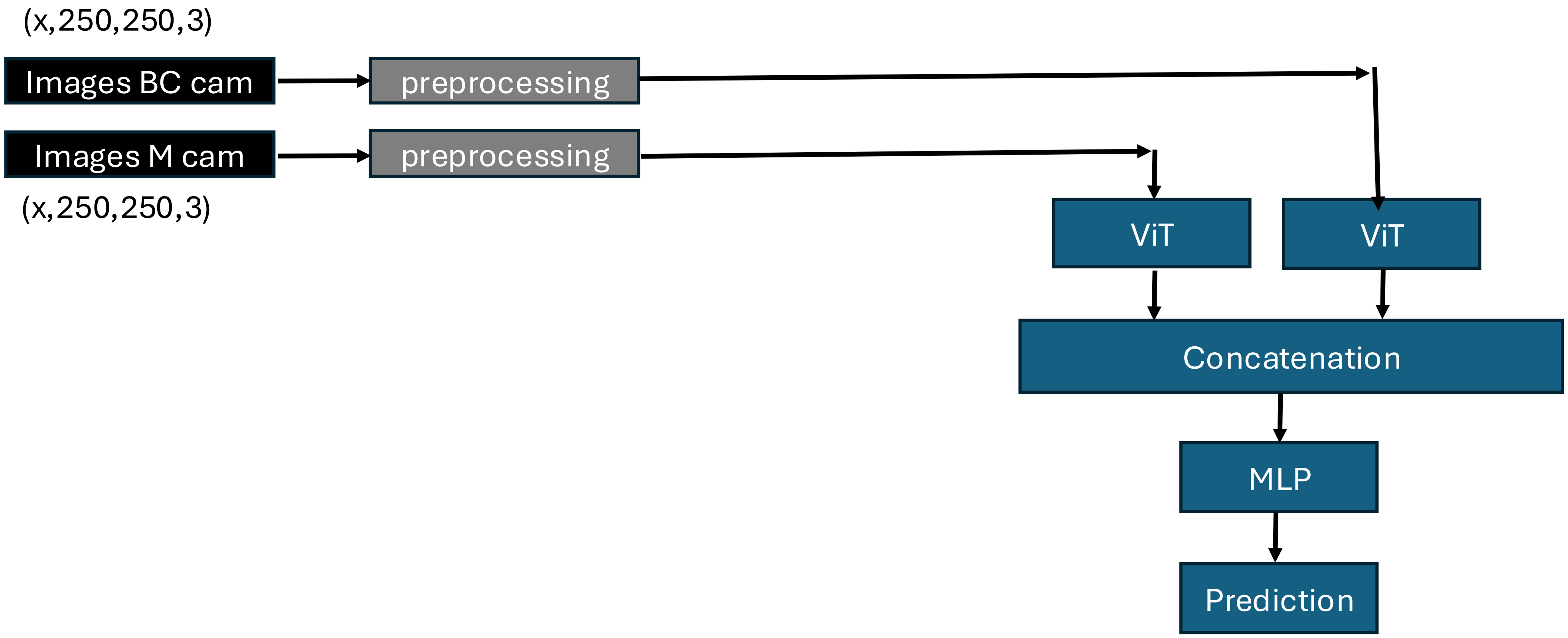
Group 17

Garance Boesinger, Daniel Elmaleh, Nathalie Künzle

Review from last time

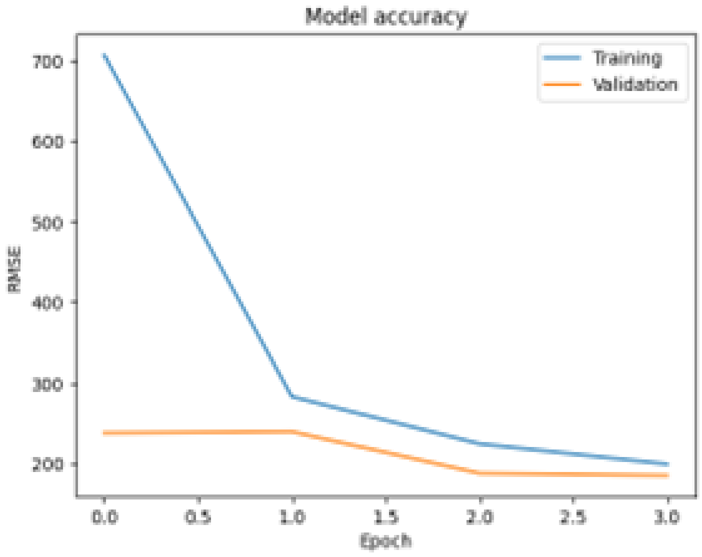
- Corrected selection of meteo + clear sky data times
- Min-max normalisation of all input data

Review: best Model from last time



Initial run

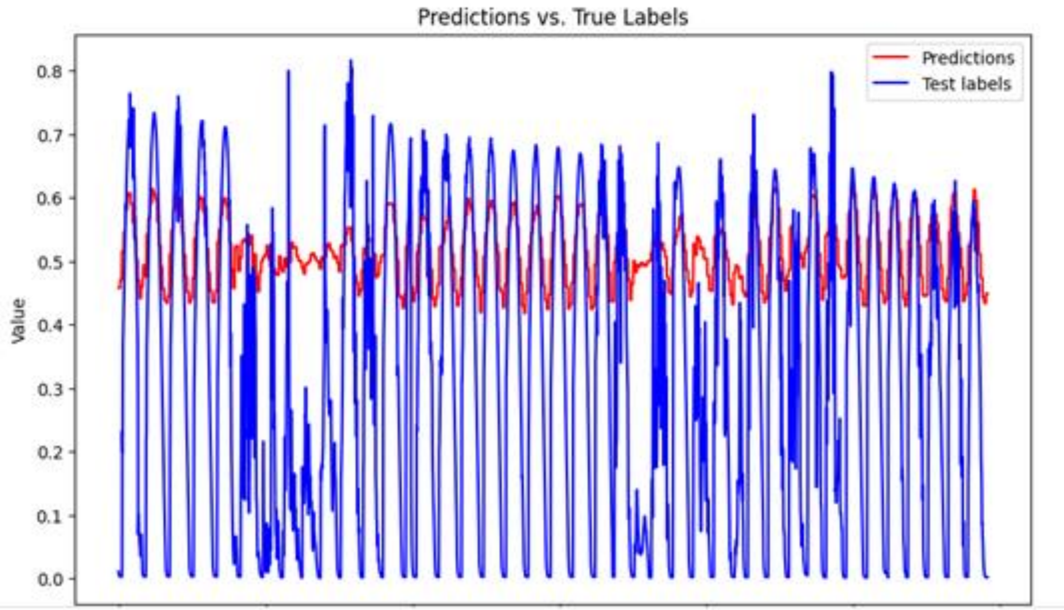
Training



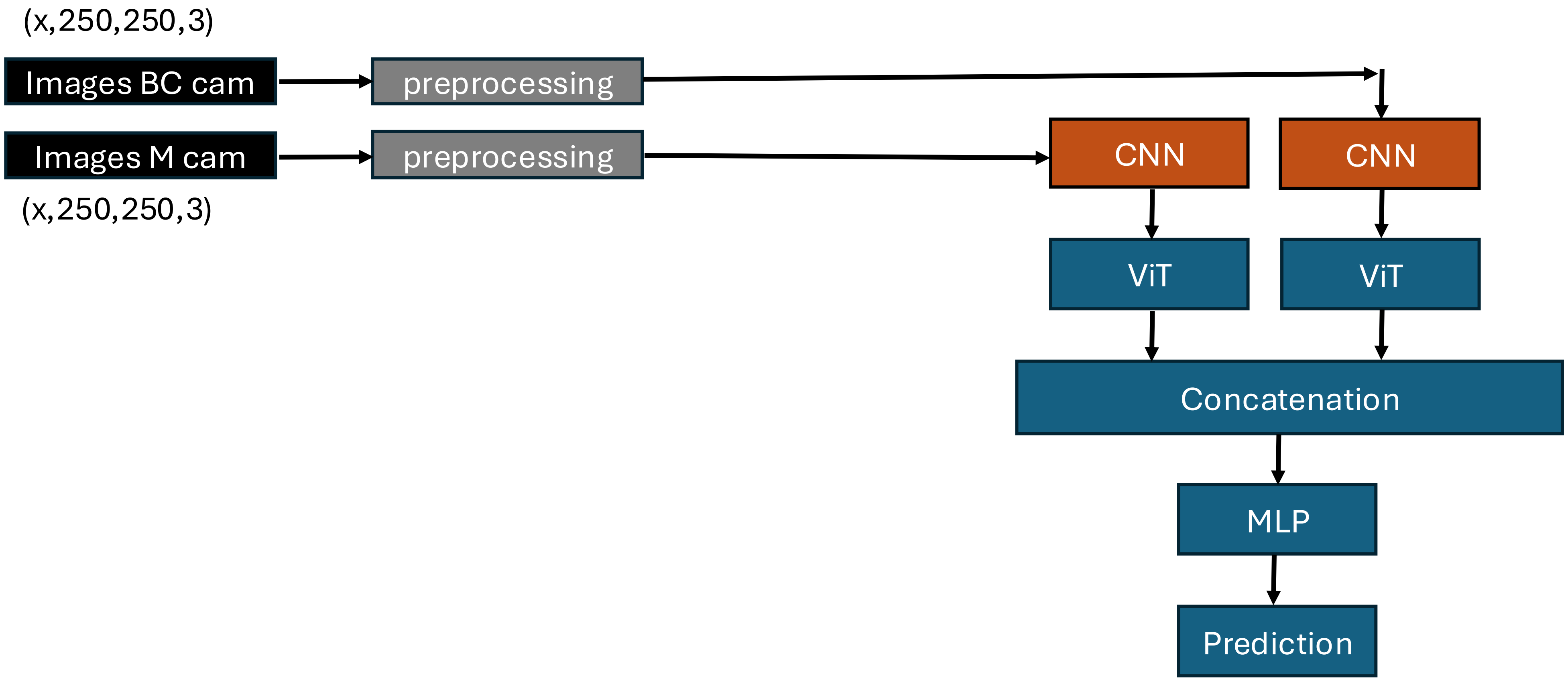
Evaluation on known test set

```
* Evaluating the performance of the trained network on the unseen test dataset *

93/93 ━━━━━━━━━━━━━━━━━━━━ 8s 38ms/step - loss: 0.1675 - mse: 0.1675

Error - root_mean_squared_error: 247.400
Loss - mean_squared_error: 0.058
```

RMSE = 247

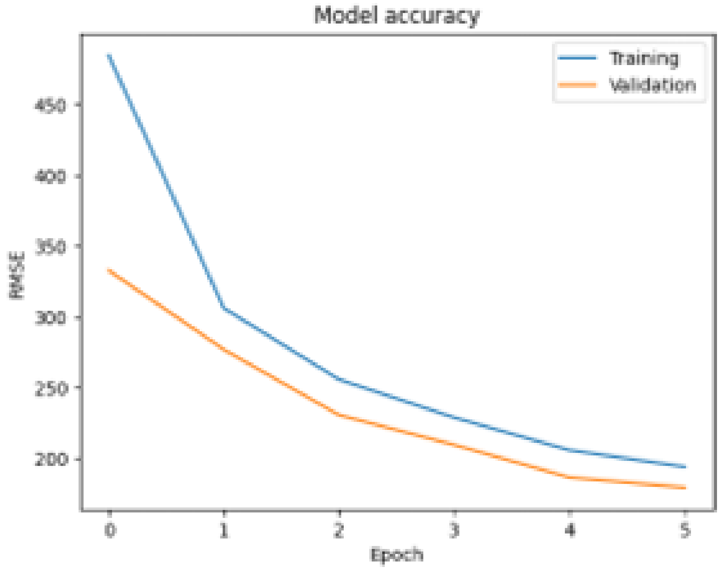


Added CNN layer



With CNN layer

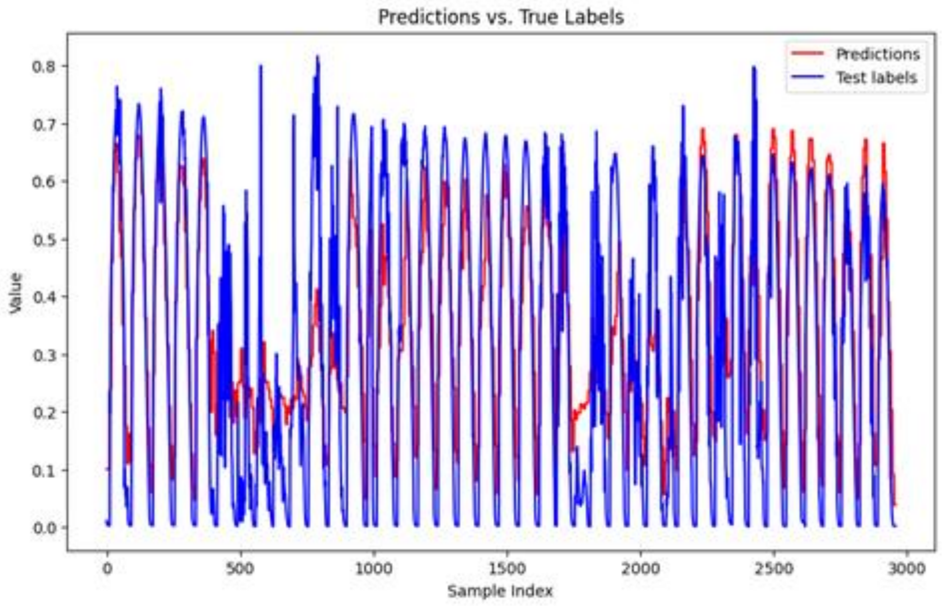
Training



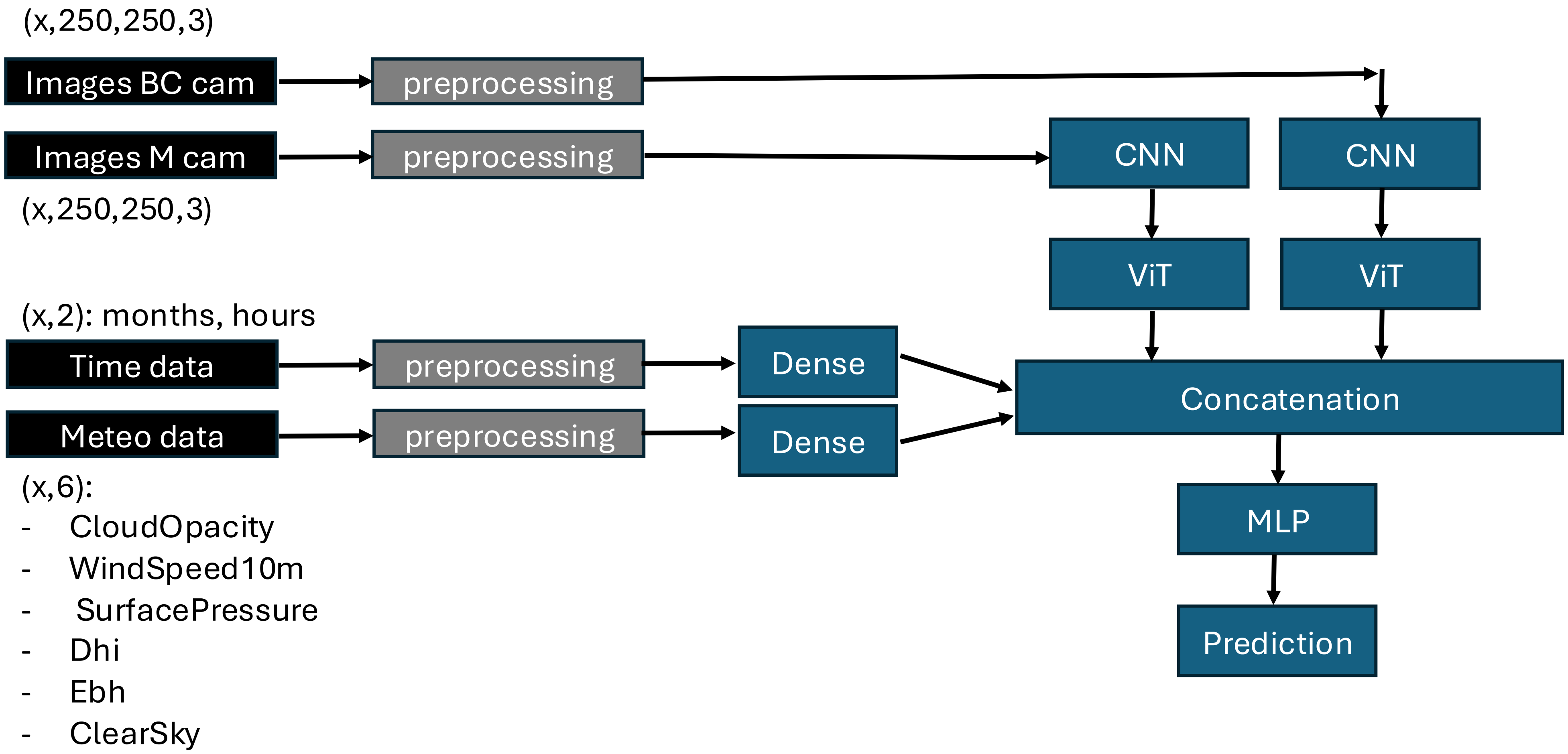
Evaluation on known test set

```
* Evaluating the performance of the trained network on the unseen test dataset *

93/93 ━━━━━━━━━━━━━━━━━━━━ 21s 174ms/step - loss: 0.0468 - mse: 0.0468

Error - root_mean_squared_error: 163.842
Loss - mean_squared_error: 0.025
```

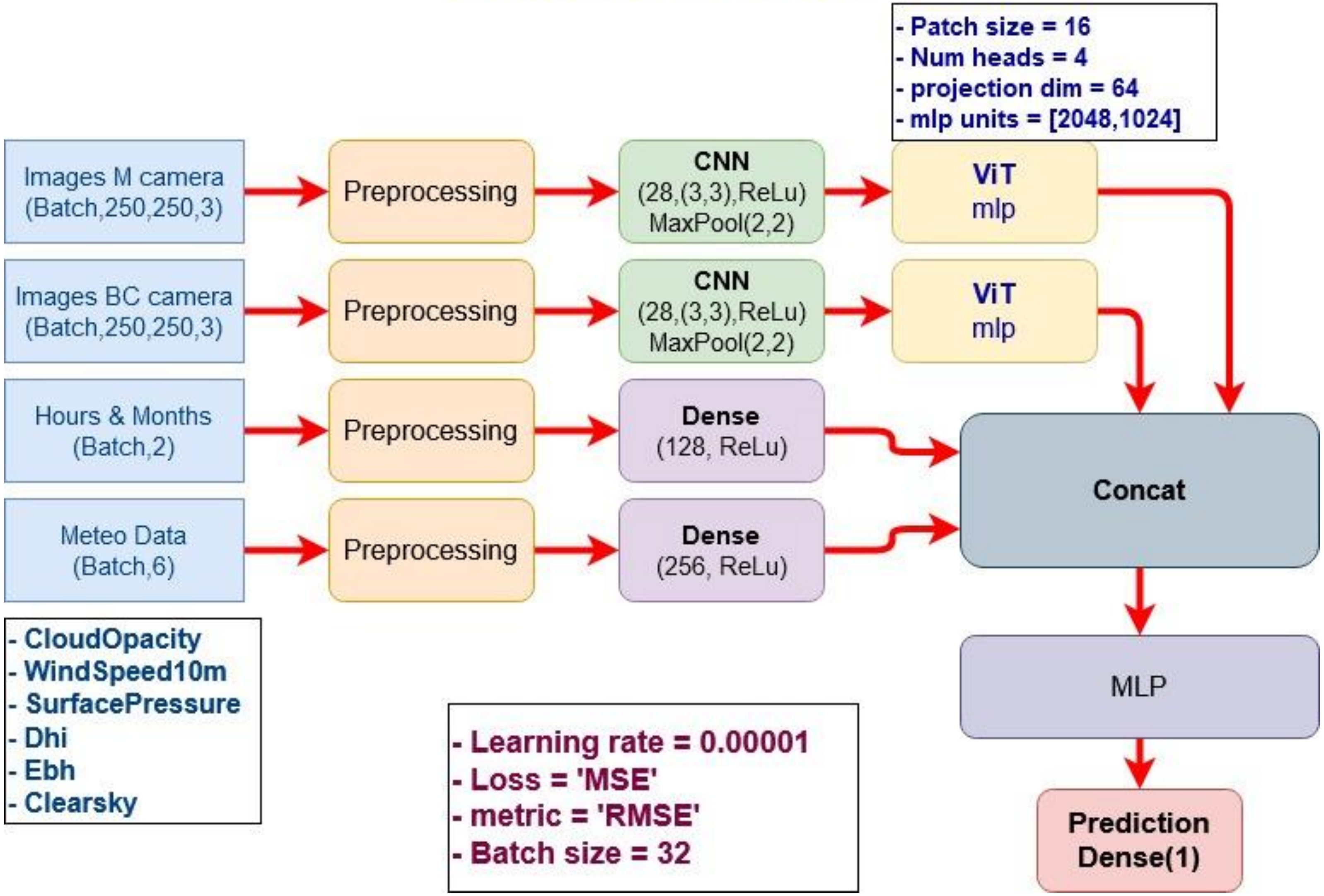
RMSE = 164



Selected model

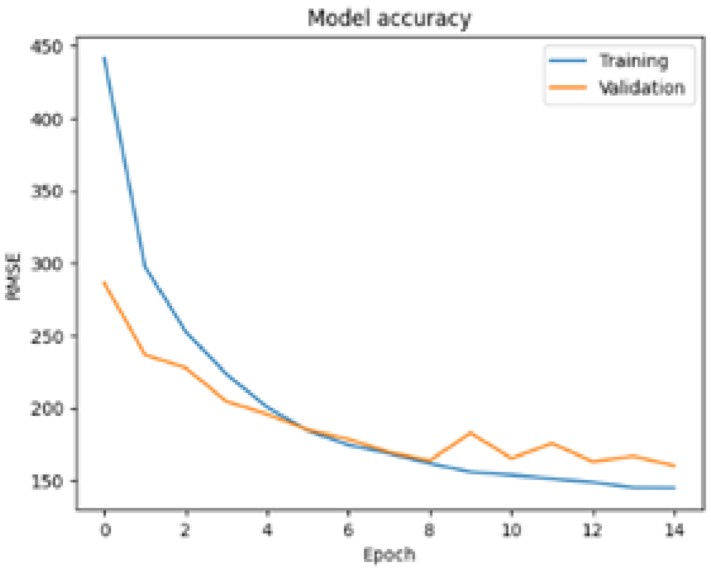


Selected Model



With meteo data inputs

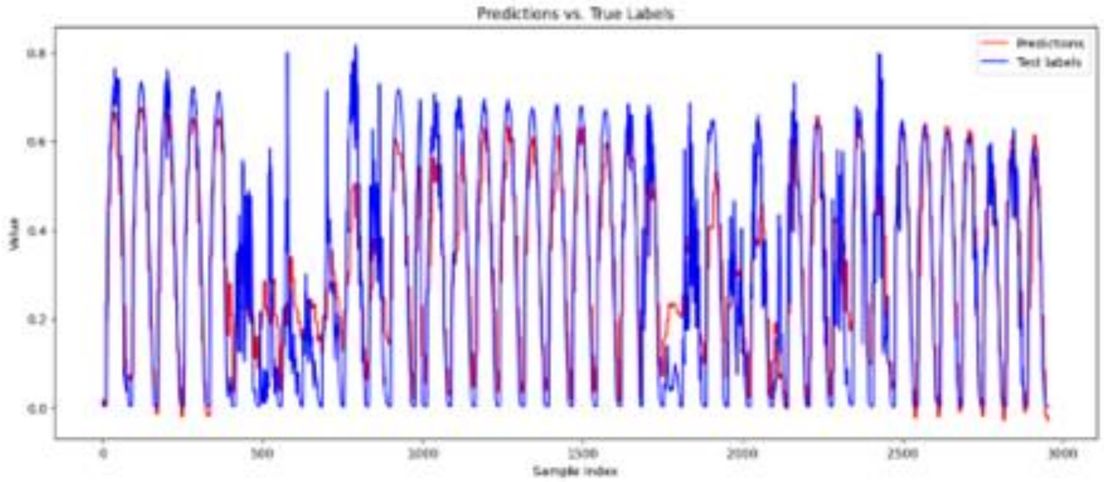
Training



Evaluation on known test set

```
* Evaluating the performance of the trained network on the unseen test dataset *

93/93 ━━━━━━━━━━━━━━━━━━━━ 20s 171ms/step - loss: 0.0253 - mse: 0.0253

Error - root_mean_squared_error: 120.585
Loss - mean_squared_error: 0.014
```

RMSE = 120

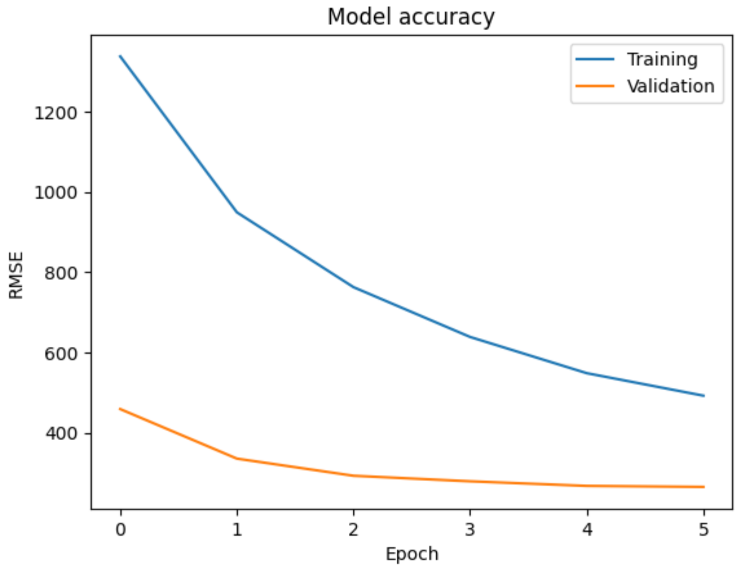


Model Optimizations

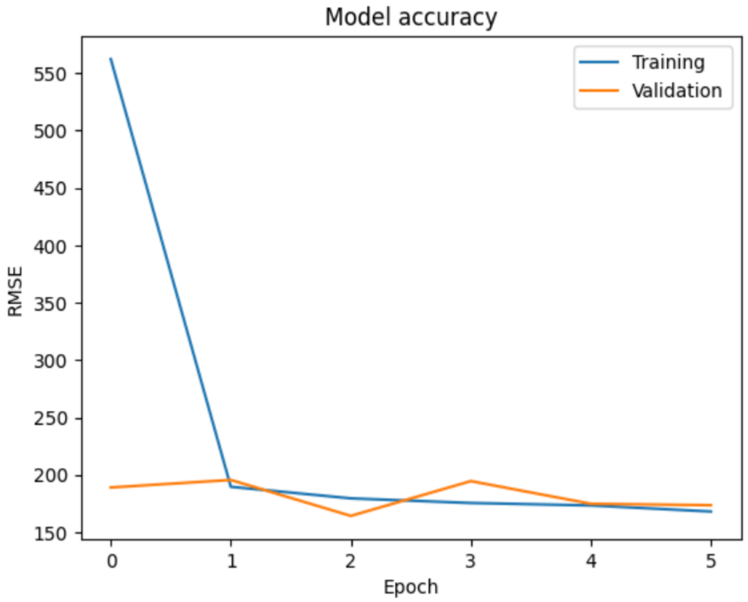
- With / without CNN layer
- CNN layer parameter
- MLP layer parameter
- Dense layers
- Learning rate
- ReduceLROnPlateau & EarlyStopping

Parameter search : learning rate

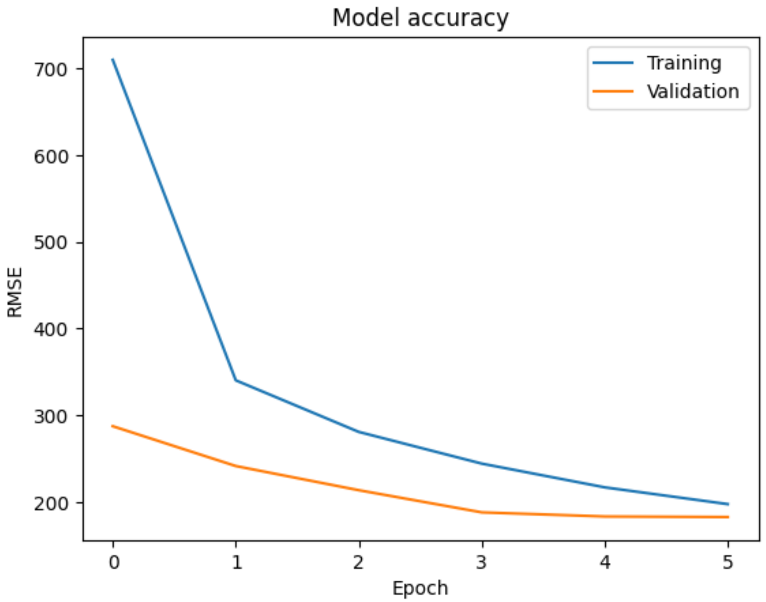
Lr = 0.00001



Lr = 0.0001 -> Best score

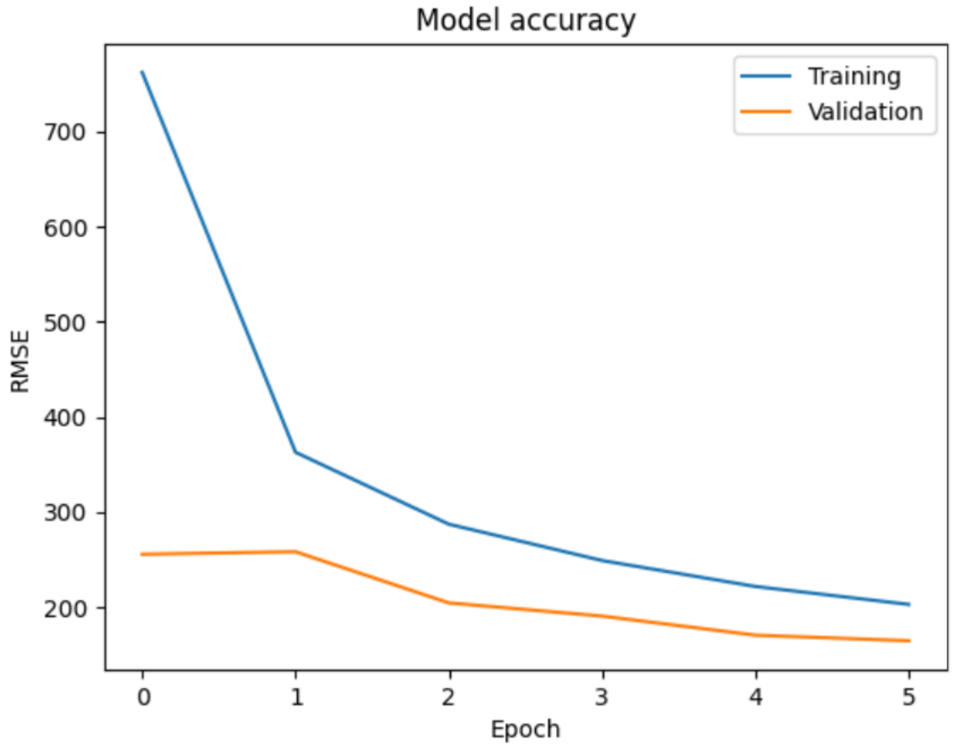


Lr = 0.001

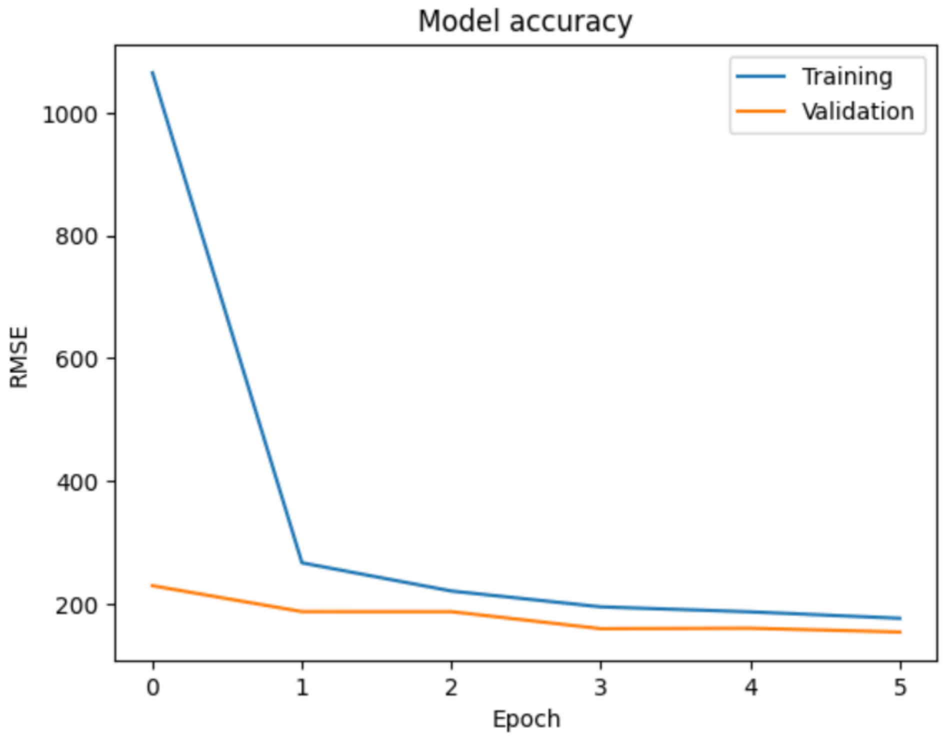


Parameter search

Increase nb of transformer

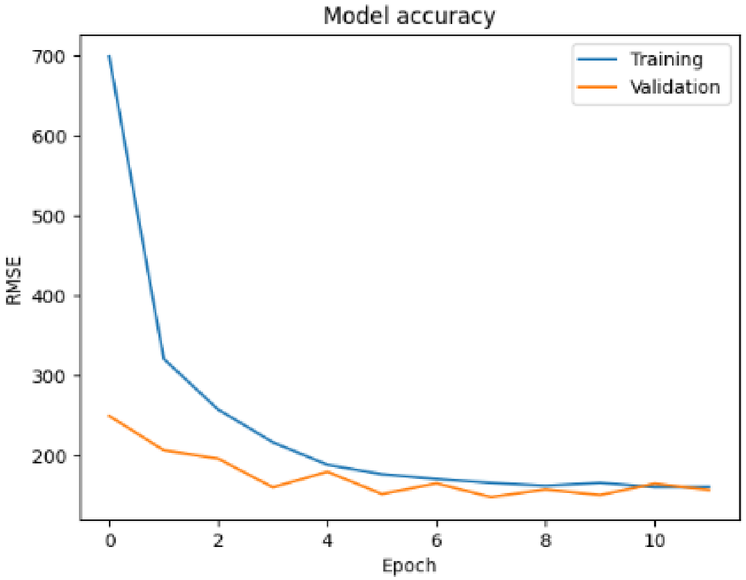


Reduced patch size (16 to 5)



Submission

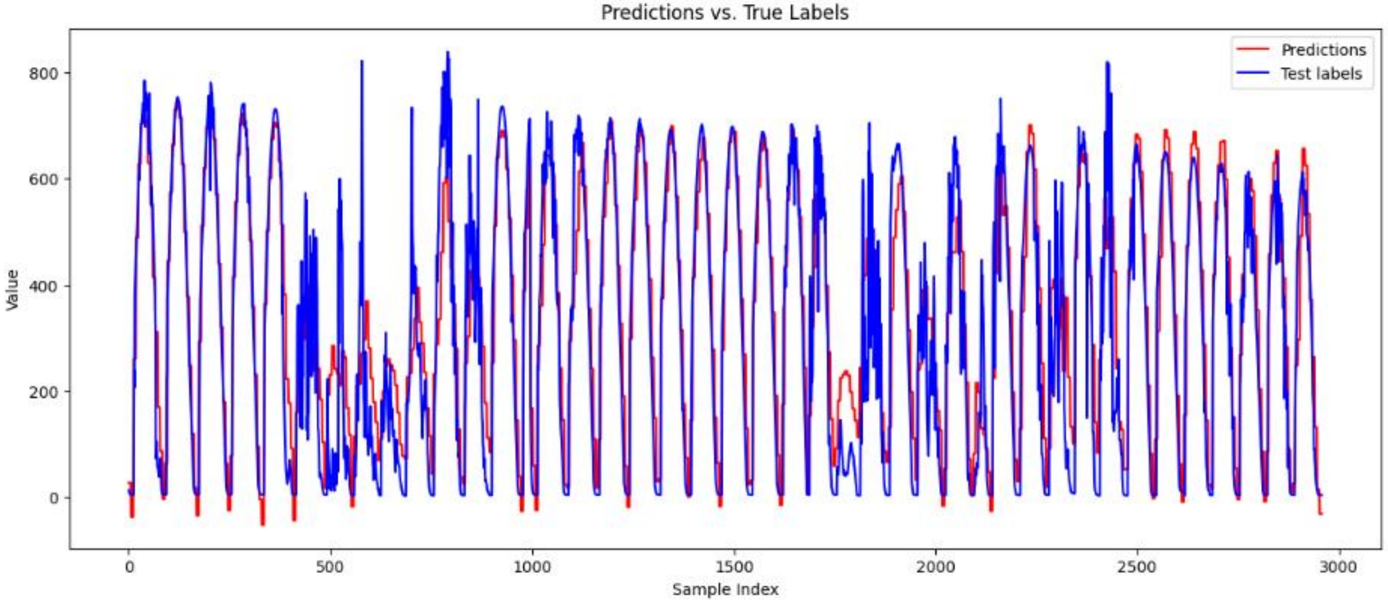
Training



Evaluation on known test set

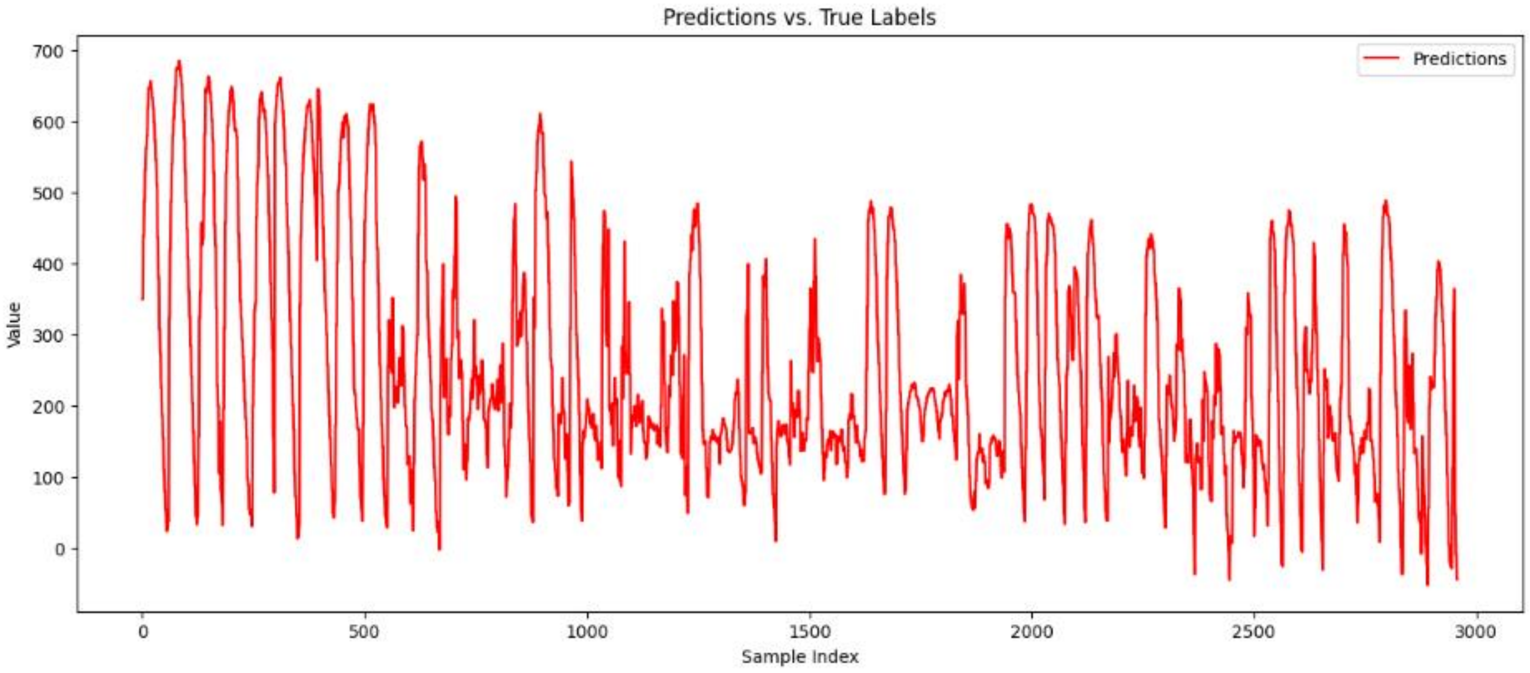
```
* Evaluating the performance of the trained network on the unseen test dataset *

93/93 ━━━━━━━━━━━━━━━━━━━━ 22s 182ms/step - loss: 0.0185 - mse: 0.0185

Error - root_mean_squared_error: 107.129
Loss - mean_squared_error: 0.011
```

RMSE = 107



Submission

Prediction for final test set



RMSE = 106.65