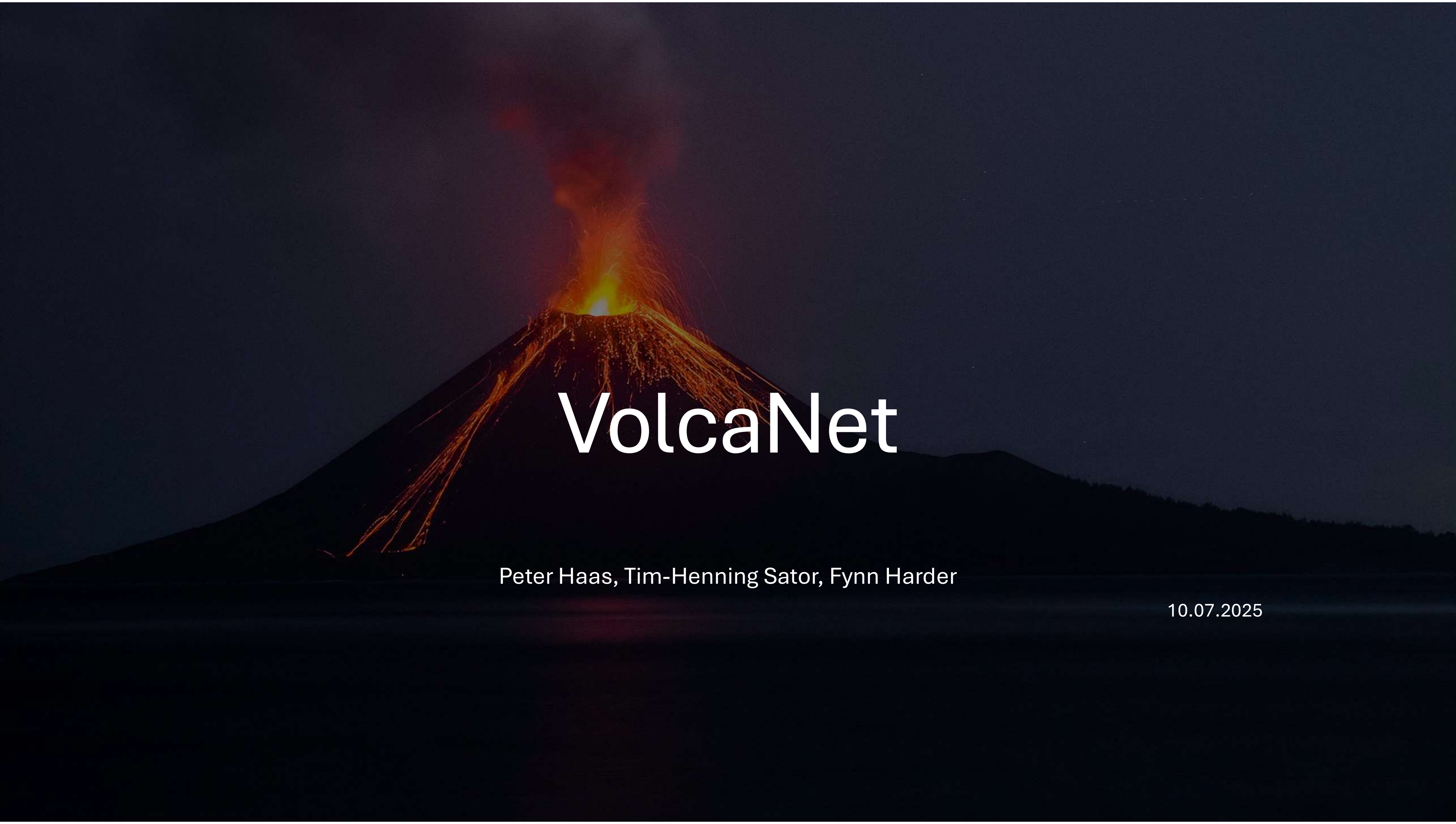


VolcaNet

Peter Haas, Tim-Henning Sator, Fynn Harder

10.07.2025

Introduction

- Exploring interesting topics in the field of satellite images and machine learning
- Peters research background
- Original idea: forecasting of volcanic eruptions
- Due to time constraint: focusing on volcano monitoring
- Led us to review recent research, especially CNNs on **satellite imagery** for **volcanic activity detection**

Literature Review

- Amato, Eleonora, et al. "**A Deep convolutional neural network for detecting volcanic thermal anomalies from satellite images**." Remote Sensing 15.15 (2023): 3718.
- Automatic detection algorithm to find **volcanic thermal anomalies in satellite images**
- Infrared data from different satellites (Data: 100/100)
- Transfer learning: **fine-tuned a pre-trained SqueezeNet CNN**

Dataset

- **Source**: European Space Agency's **Sentinel-2 mission**
- **Sentinel-2A, Sentinel-2B** (revisit frequency 5 days)
- Captures **high-resolution images of the Earth's surface**. Used to monitor forests, agriculture, water, and volcanoes.
- Focus on **specific image bands** from Sentinel-2:
  - **B8A (near-infrared)**
  - **B11 (shortwave infrared 1)**
  - **B12 (shortwave infrared 2)**
- Useful because **infrared light is very sensitive to heat and surface changes**
- Combining these bands as a **false color image**, we can highlight **hot spots** and **surface features** linked to volcanic activity

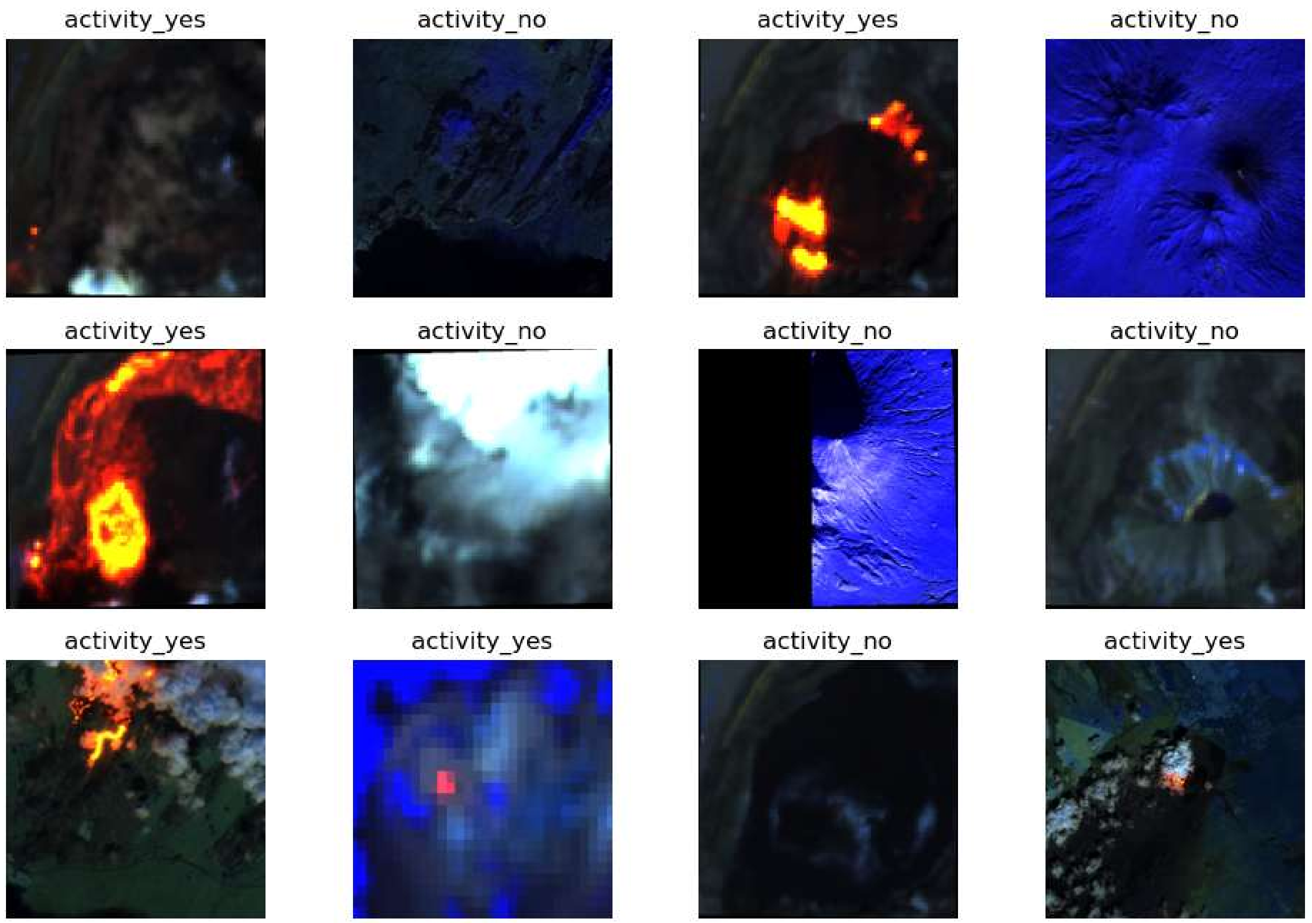
Dataset

- **Approach - Create an own dataset:**
  - Filter for volcanic eruptions from 2016 on (Smithsonian Institute)
  - Get all images 365 days before and after eruption date from Google Earth Engine
  - Sort images by hand in showing „volcanic actvity“ and showing no „volcanic activity“
  - Balancing data in 50/50 ratio
  - Resulting in a dataset of 830 (415/415) images of 9 volcanoes
  - Enlarging the original dataset by creating synthetic satellite images of both states by using common augmenting techniques (rotation, contrast/brightness adaption, blurring, ...) $\rightarrow$ 4150 images (2075/2075)

Dataset

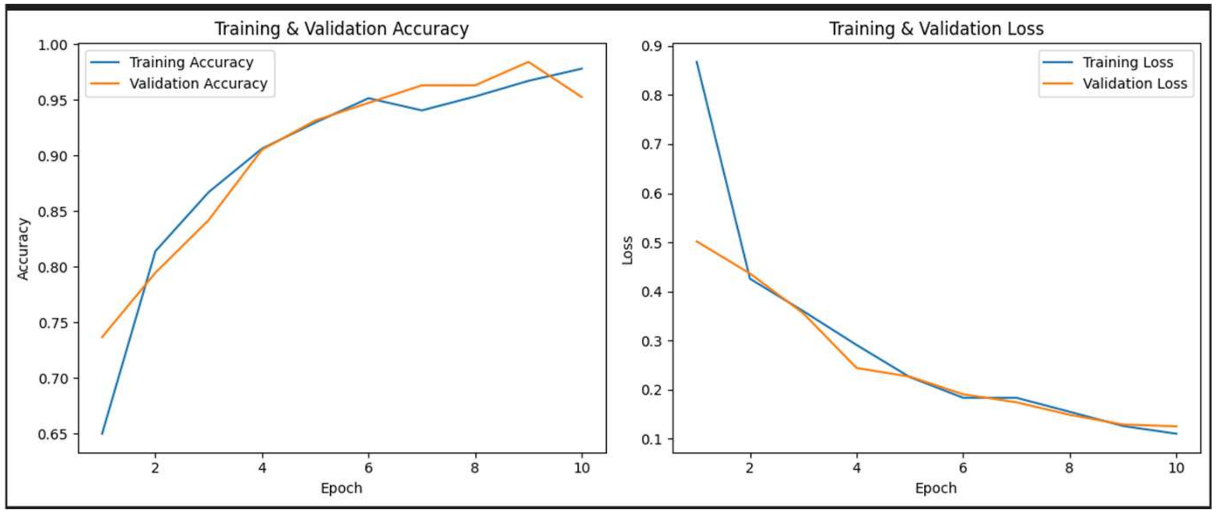
Volcanoes:

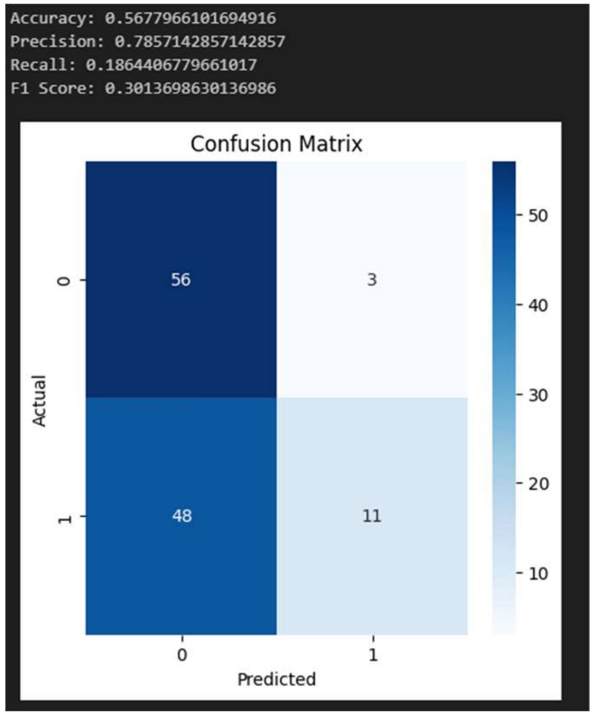
- Bezymianny
- Etna
- Home Reef
- Kilauea
- Mauna
- Mayon
- Merapi
- Popocatepetl
- Stromboli
- Barren Island (out-of-sample)



Baseline Model

- Small 1-layer CNN for the original data set (830 images)



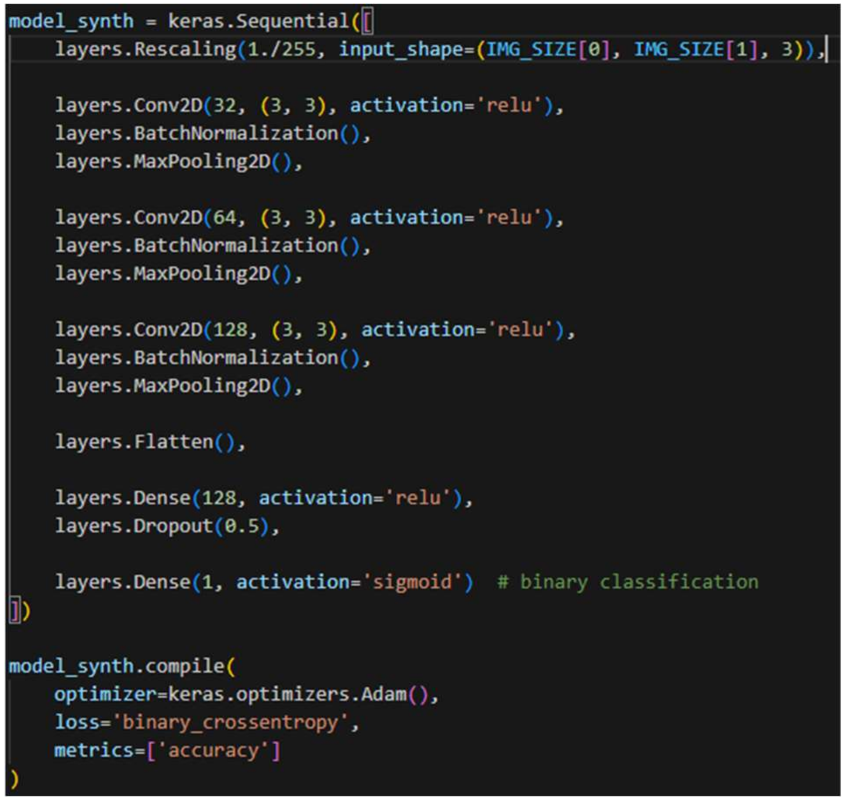


```
Accuracy: 0.5677966101694916
Precision: 0.7857142857142857
Recall: 0.1864406779661017
F1 Score: 0.3013698630136986
```

Model Definition and Evaluation

Deep Convolutional Neural Network (CNN) on enlarged dataset

- **3 convolutional blocks** (Conv2D) with **increasing filter sizes** (32 → 64 → 128)
- Each Conv block is followed by **Batch Normalization** to stabilize training and **Max Pooling** to **reduce spatial dimensions**
- A Flatten layer to convert the 3D feature maps to a 1D vector
- A Dense layer with 128 units and Dropout (0.5) to reduce overfitting

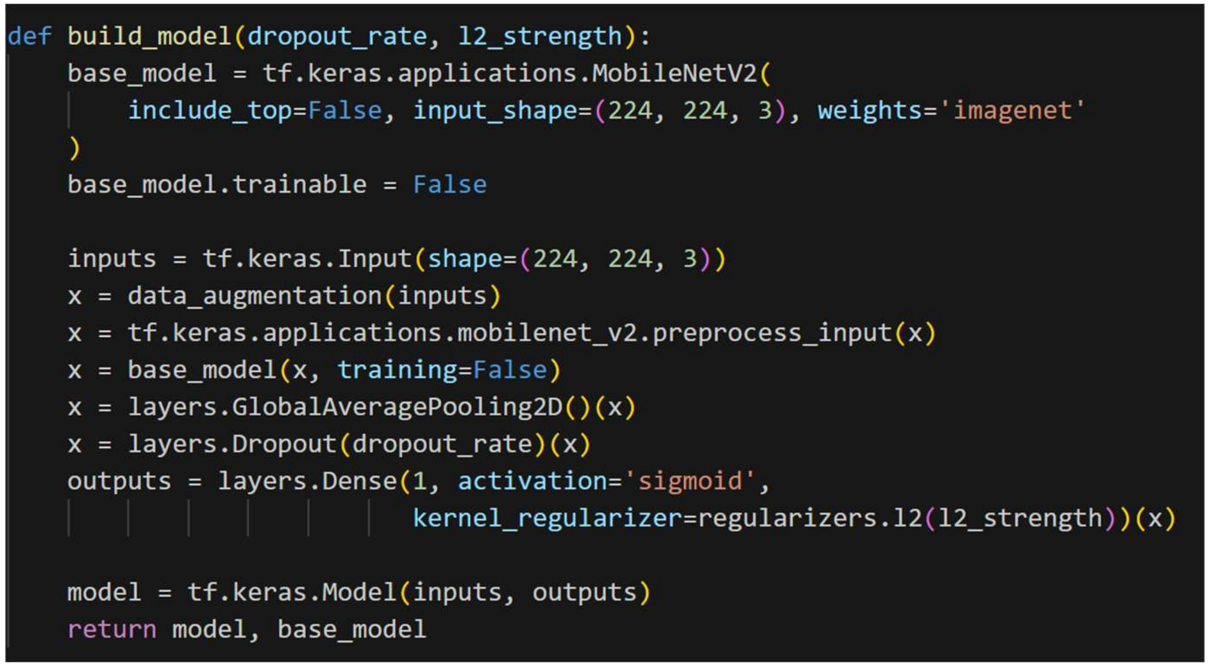


```python
model_synth = keras.Sequential([
    layers.Rescaling(1./255, input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3)),

    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),

    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),

    layers.Flatten(),

    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),

    layers.Dense(1, activation='sigmoid')  # binary classification
])

model_synth.compile(
    optimizer=keras.optimizers.Adam(),
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

Model Definition and Evaluation

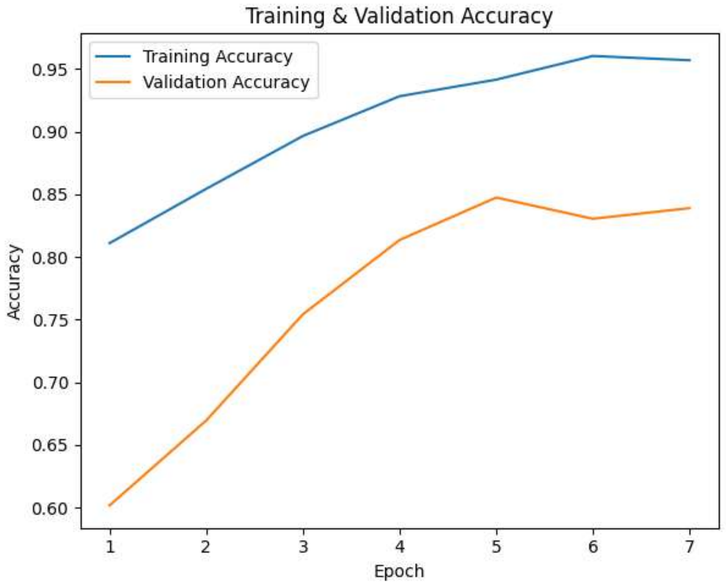
Transfer Learning: **MobileNetV2 on enlarged dataset**

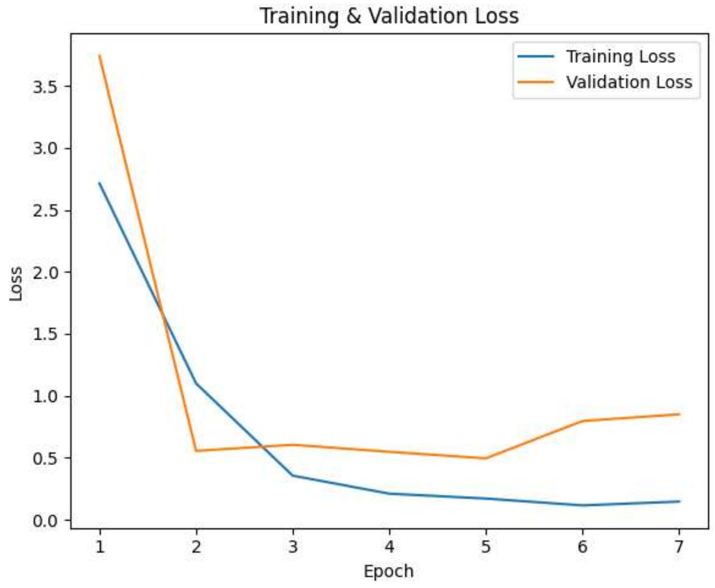
- **Pretrained Convolutional Neural Network** originally trained on **Images** (ImageNet)
- **GlobalAveragePooling2D** layer to compress the extracted features
- **Dropout layer** to reduce overfitting
- Final Dense layer with **sigmoid activation** for binary classification and **L2 regularization** (0.001) to further penalize overly complex weights

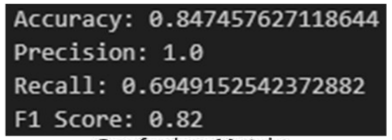


```python
def build_model(dropout_rate, l2_strength):
    base_model = tf.keras.applications.MobileNetV2(
        include_top=False, input_shape=(224, 224, 3), weights='imagenet'
    )
    base_model.trainable = False

    inputs = tf.keras.Input(shape=(224, 224, 3))
    x = data_augmentation(inputs)
    x = tf.keras.applications.mobilenet_v2.preprocess_input(x)
    x = base_model(x, training=False)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dropout(dropout_rate)(x)
    outputs = layers.Dense(1, activation='sigmoid',
                           kernel_regularizer=regularizers.l2(l2_strength))(x)

    model = tf.keras.Model(inputs, outputs)
    return model, base_model
```

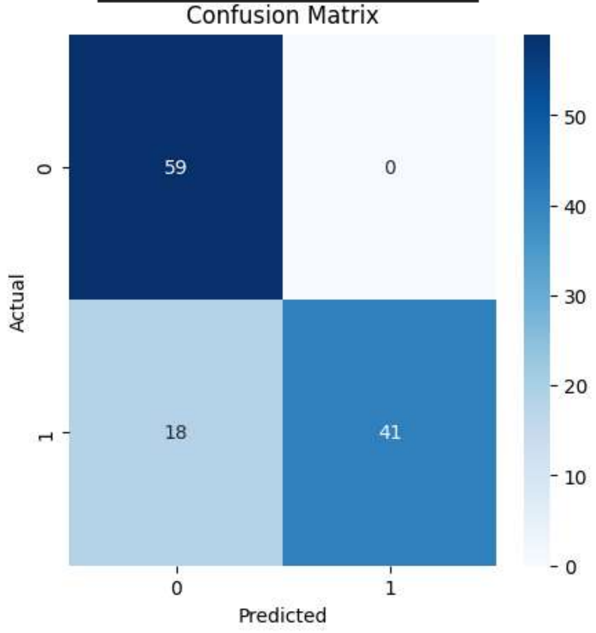
Results - CNN



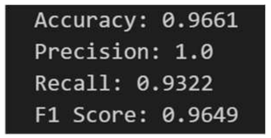




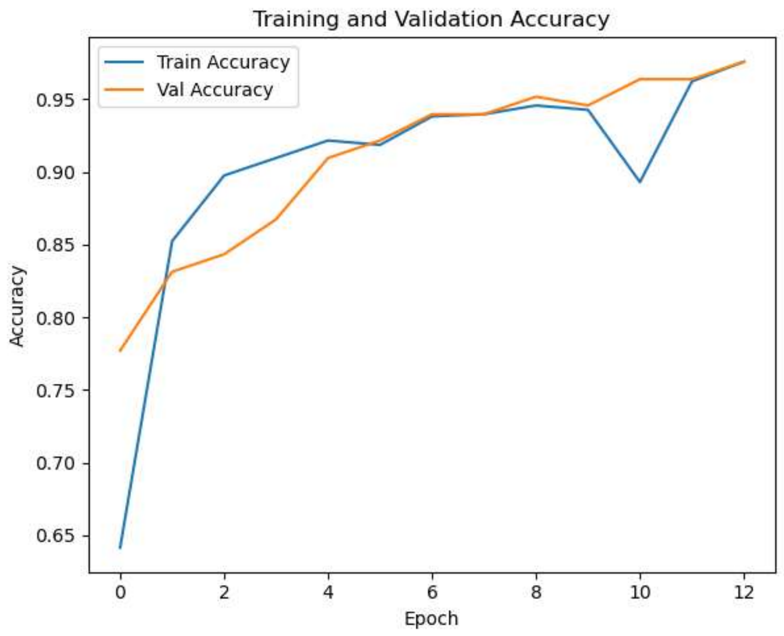
```
Accuracy: 0.847457627118644
Precision: 1.0
Recall: 0.6949152542372882
F1 Score: 0.82
```

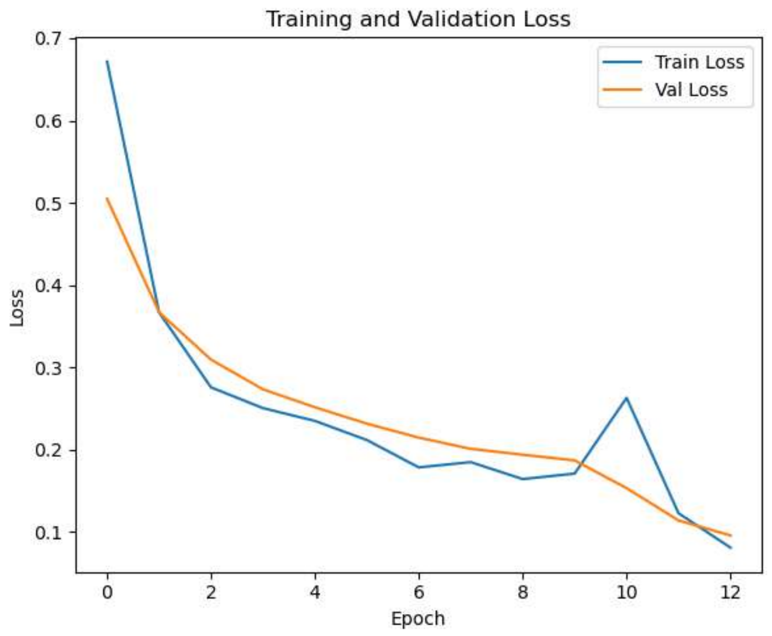


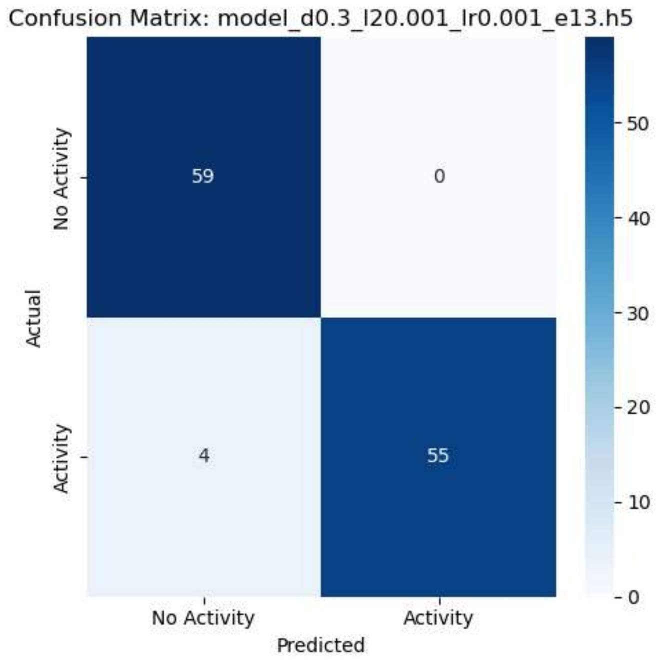
Results - MobileNet



```
Accuracy: 0.9661
Precision: 1.0
Recall: 0.9322
F1 Score: 0.9649
```


Training and Validation Accuracy
Train Accuracy
Val Accuracy
Accuracy
Epoch


Training and Validation Loss
Train Loss
Val Loss
Loss
Epoch


Confusion Matrix: model_d0.3_l20.001_lr0.001_e13.h5
Actual
No Activity
Activity
Predicted
No Activity
Activity