Heart Disease Detection Using ECG Signals

DATASET UESD

1. mit-bih-arrhythmia-database-1.0.0
2. mit-bih-artial-fibrillation-database-1.0.0
3. mit-bih-malignant-ventricular-ectopy-database-1.0.0
4. mit-bih-normal-sinus-rhythm-database-1.0.0
5. mit-bih-st-change-database-1.0.0

MIT-BIH-ARRHYTHMIA-DATABASE-1.0.0

- Disease Focus: Cardiac Arrhythmias
- Description: This dataset contains ECG signals from patients with various types of arrhythmias — abnormal heart rhythms that may be too fast, too slow, or irregular. Some common arrhythmias detectable in this dataset include:
    1. Premature Ventricular Contractions (PVCs)
    2. Bundle Branch Block
    3. Atrial Premature Beat
    4. Supraventricular Tachycardi
    5. Ventricular Tachycardia (VT)

MIT-BIH-ARTIAL-FIBRILLATION-DATABASE-1.0.0

- Disease Focus: Atrial Fibrillation (AF)
- Description: This dataset is specifically curated to help detect Atrial Fibrillation, a common type of arrhythmia where the atria (upper heart chambers) beat chaotically and out of sync with the ventricles. It can lead to:
    1. Stroke
    2. Heart Failure
    3. Fatigue and Palpitations

MIT-BIH-MALIGNANT-VENTRICULAR-ECTOPY-DATABASE-1.0.0

- Disease Focus: Ventricular Arrhythmias (Life-threatening)
- Description: This dataset includes ECG signals from patients who had severe ventricular arrhythmias such as:
    1. Ventricular Tachycardia (VT)
    2. Ventricular Fibrillation (VF)
    3. Multiform Premature Ventricular Complexes (PVCs)
    4. Causing sudden Cardiac death

MIT-BIH-NORMAL-SINUS-RHYTHM-DATABASE-1.0.0

- This dataset contains ECG recordings from healthy individuals with normal sinus rhythm. It serves as a baseline to distinguish between:
  1. Normal cardiac activity
  2. Abnormal or diseased conditions
- Models trained on both normal and abnormal data can learn to detect deviations from healthy heart function.

MIT-BIH-ST-CHANGE-DATABASE-1.0.0

- Disease Focuses: Myocardial Ischemia (ST Elevation or Depression)
- Description: ST-segment deviations are signs of reduced blood flow to the heart (ischemia). This can indicate:
    1. Acute coronary syndrome
    2. Heart attack (Myocardial Infarction)

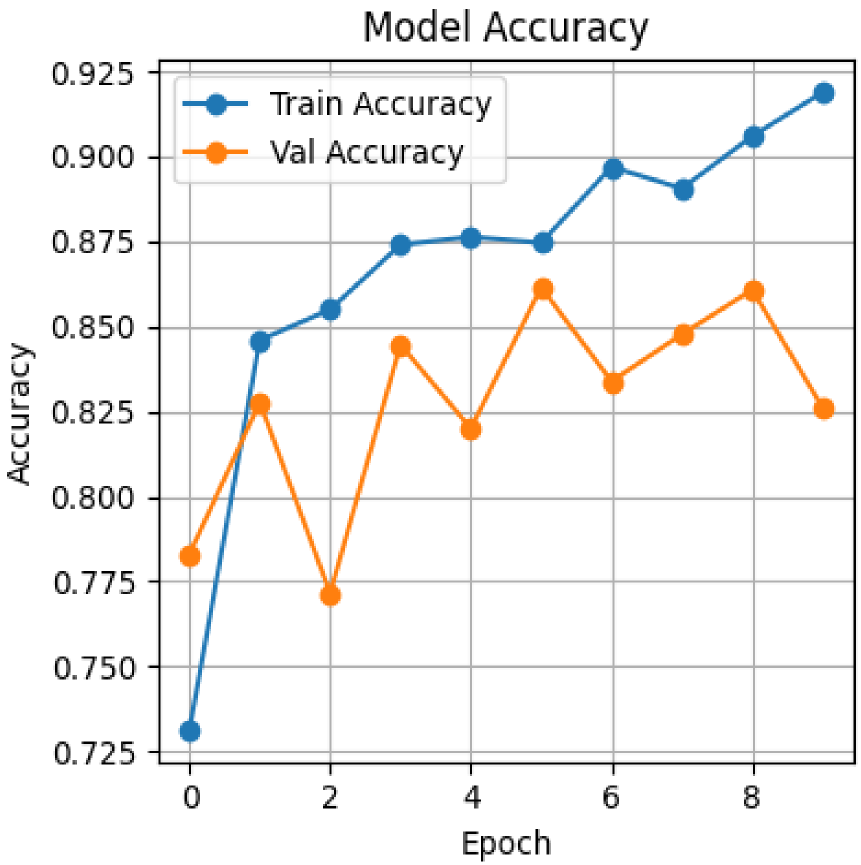
Models Used

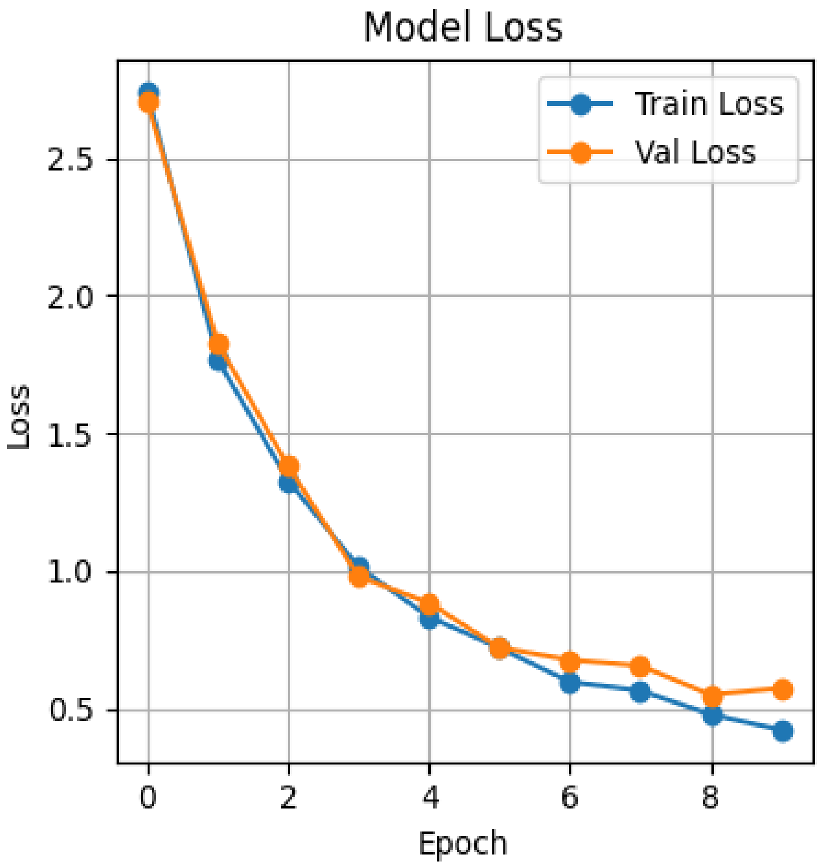
1. 2 Custom CNN Models
2. VGG16
3. ResNet50
4. ResNet101
5. Xception
6. DenseNet121

First Custom CNN model

CNN Model Architecture (Simplified)

1. **Input Layer:**
   1. Shape: (128, 128, 3)
2. **Conv Block 1:**
   1. Conv2D (32 filters, 3×3, ReLU)
   2. Batch Normalization
   3. MaxPooling (2×2)
   4. Dropout (rate = 0.3)
3. **Conv Block 2:**
   1. Conv2D (64 filters, 3×3, ReLU)
   2. MaxPooling (2×2)
4. **Fully Connected Layers:**
   1. Flatten
   2. Dense (128 units, ReLU, L2 Regularization)
   3. Dense (32 units, ReLU)
5. **Output Layer:**
   1. Dense (5 units, Softmax)



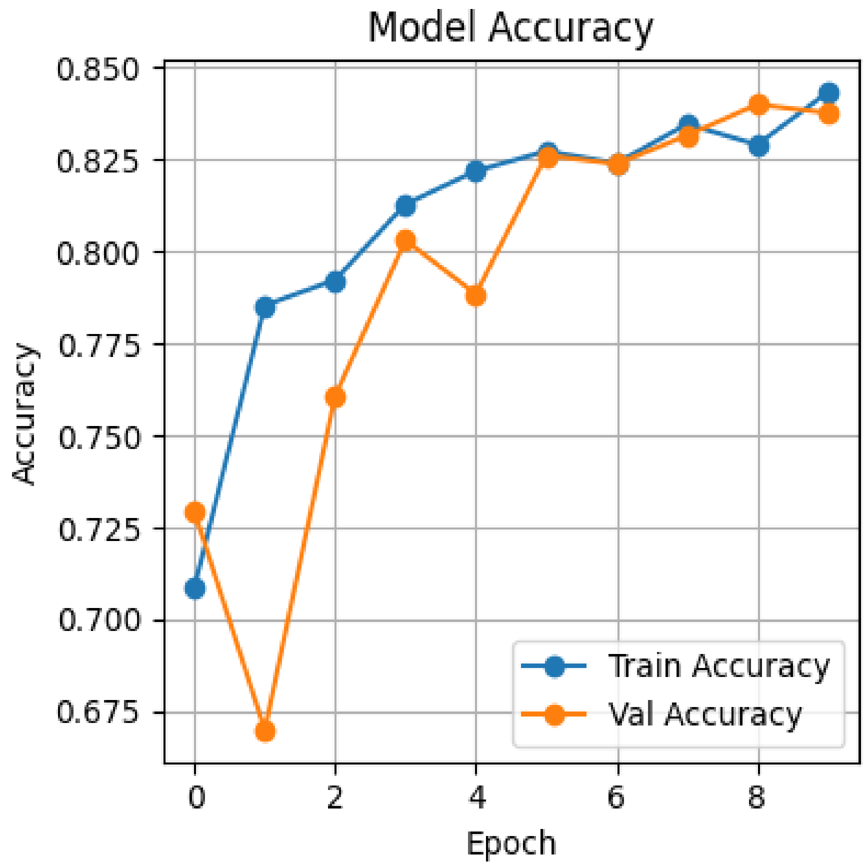


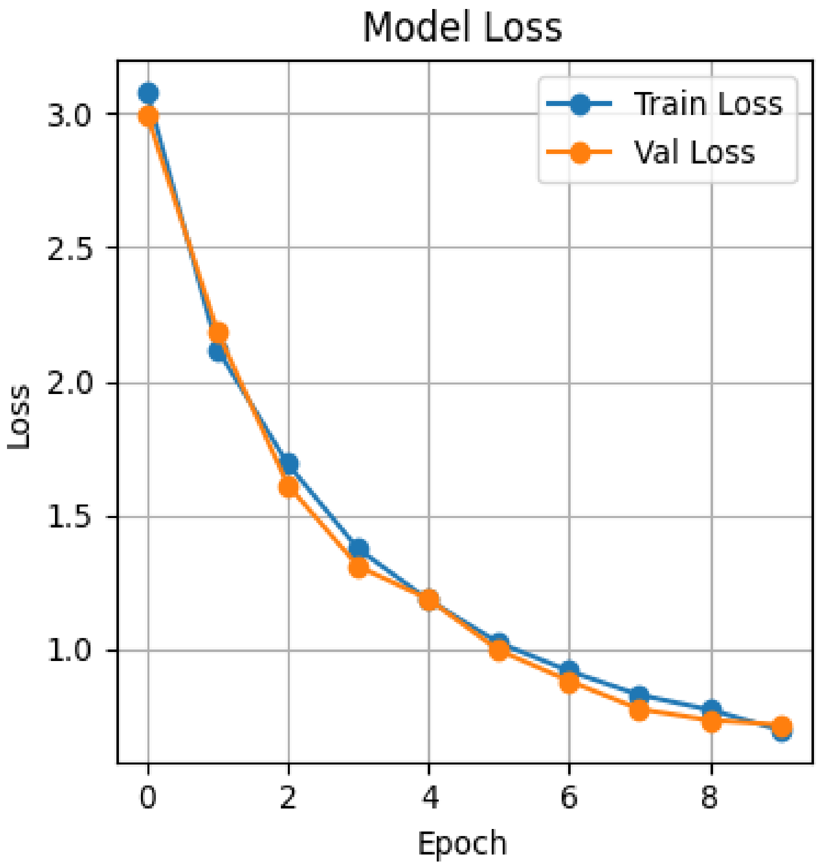
Validation Set Accuracy: 82.62%

Second Custom CNN model

CNN Model Architecture (Simplified)

1. **Input Layer:**
    1. Shape: (128, 128, 3)
2. **Conv Block 1:**
    1. Conv2D (32 filters, 3×3, ReLU)
    2. Batch Normalization
    3. MaxPooling (2×2)
    4. Dropout (rate = 0.3)
1. **Data Augmentation:**
    1. Random Flip
    2. Random Rotation (±10%)
    3. Random Zoom (±10%)
2. **Conv Block 2:**
    1. Conv2D (64 filters, 3×3, ReLU)
    2. MaxPooling (2×2)
3. **Fully Connected Layers:**
    1. Flatten
    2. Dense (128 units, ReLU, L2 Regularization)
    3. Dense (32 units, ReLU, L2 Regularization)
4. **Output Layer:**
    1. Dense (5 units, Softmax)



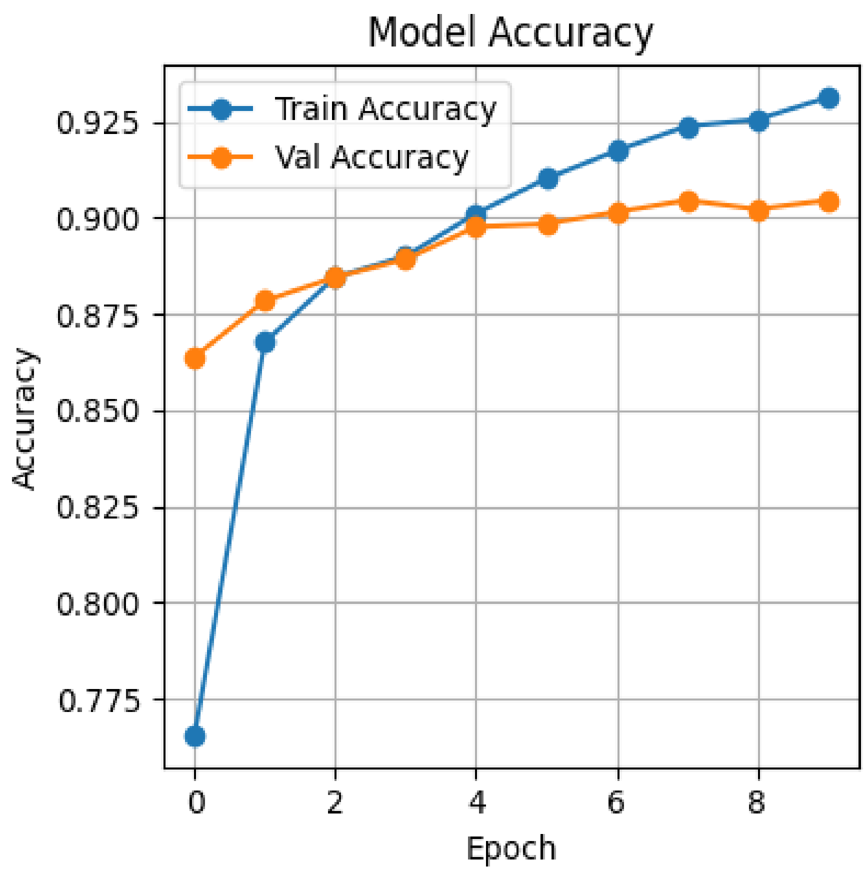


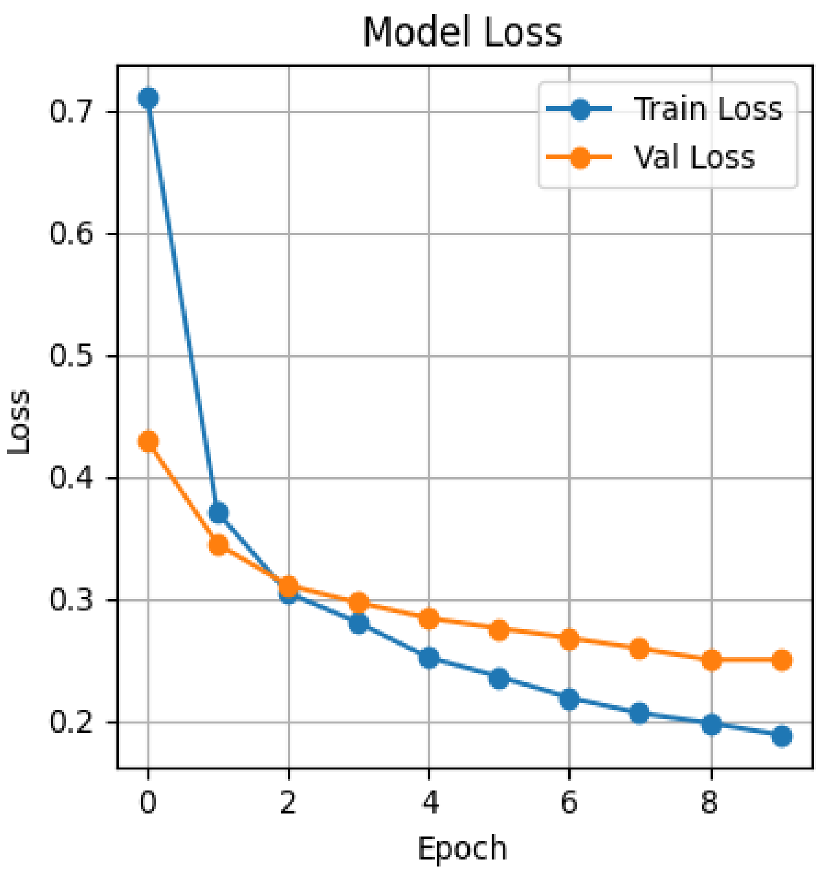
Validation Set Accuracy: 83.77%

VGG16 (transfer learning)

Transfer Learning Model Architecture

1. **Base Model:**
    1. Pretrained VGG16
    2. Weights: *ImageNet*
    3. Top layers removed (include_top=False)
    4. Input shape: *(128, 128, 3)*
    5. All layers frozen
2. **Custom Head:**
    1. Flatten
    2. Dense (64 units, ReLU)
    3. Output: Dense (5 units, Softmax)
3. **Compilation:**
    1. Optimizer: Adam (learning rate = 1e-4)
    2. Loss: Categorical Crossentropy
    3. Metric: Accuracy



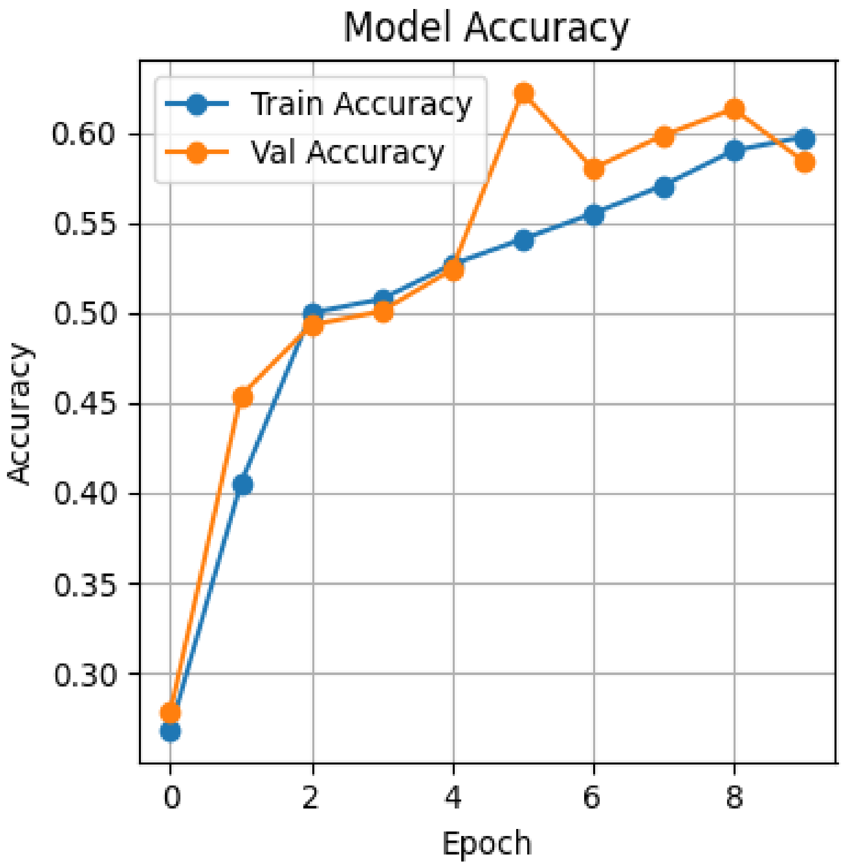


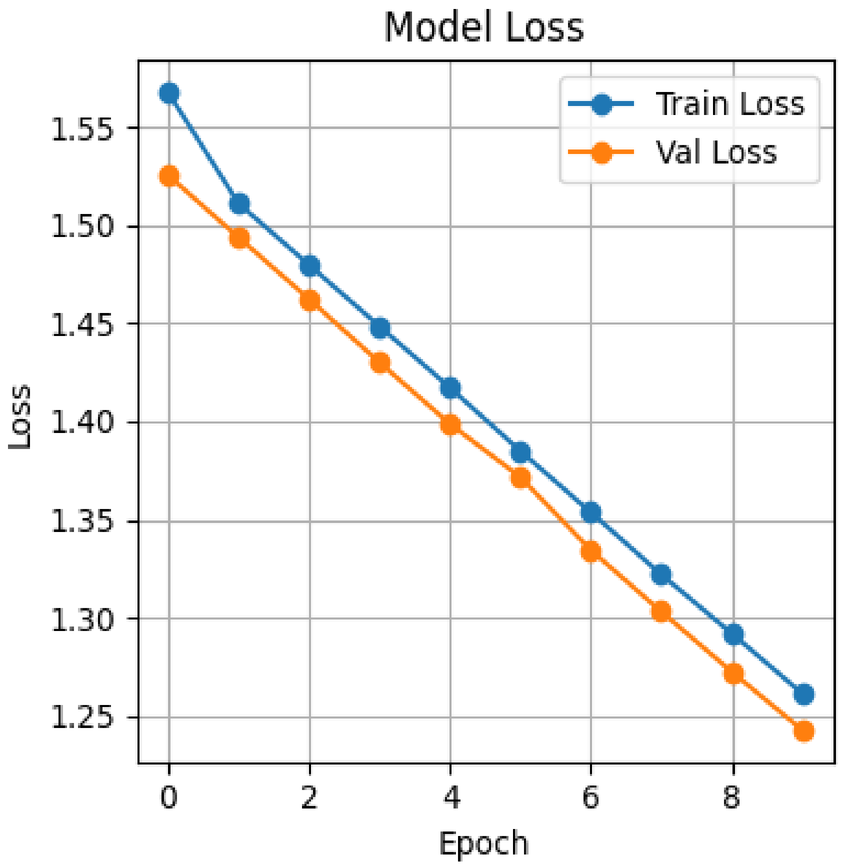
Validation Set Accuracy: 90.46%

ResNet50 (transfer learning)

Transfer Learning Model Architecture

1. **Base Model:**
    1. Pretrained ResNet50
    2. Weights: *ImageNet*
    3. Top layers removed (include_top=False)
    4. Input shape: *(128, 128, 3)*
    5. All layers frozen
2. **Custom Head:**
    1. Flatten
    2. Dense (64 units, ReLU)
    3. Output: Dense (5 units, Softmax)
3. **Compilation:**
    1. Optimizer: Adam (learning rate = 1e-4)
    2. Loss: Categorical Crossentropy
    3. Metric: Accuracy



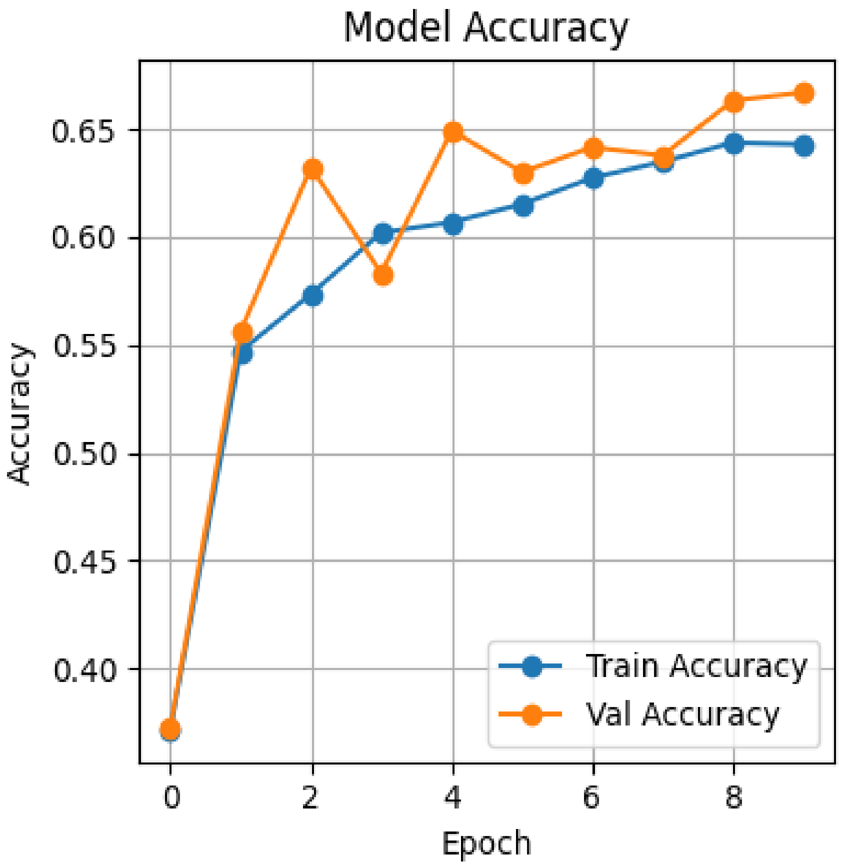


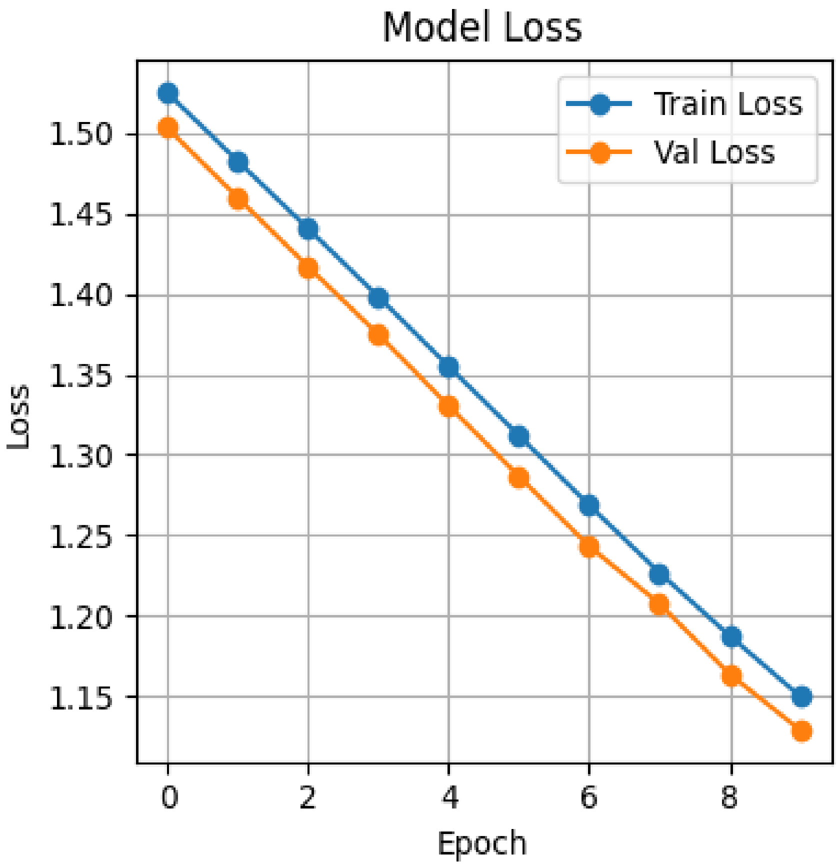
Validation Set Accuracy: 58.38%

ResNet101 (transfer learning)

Transfer Learning Model Architecture

1. **Base Model:**
    1. Pretrained ResNet101
    2. Weights: *ImageNet*
    3. Top layers removed (include_top=False)
    4. Input shape: *(128, 128, 3)*
    5. All layers frozen
2. **Custom Head:**
    1. Flatten
    2. Dense (64 units, ReLU)
    3. Output: Dense (5 units, Softmax)
3. **Compilation:**
    1. Optimizer: Adam (learning rate = 1e-4)
    2. Loss: Categorical Crossentropy
    3. Metric: Accuracy





Validation Set Accuracy: 66.69%

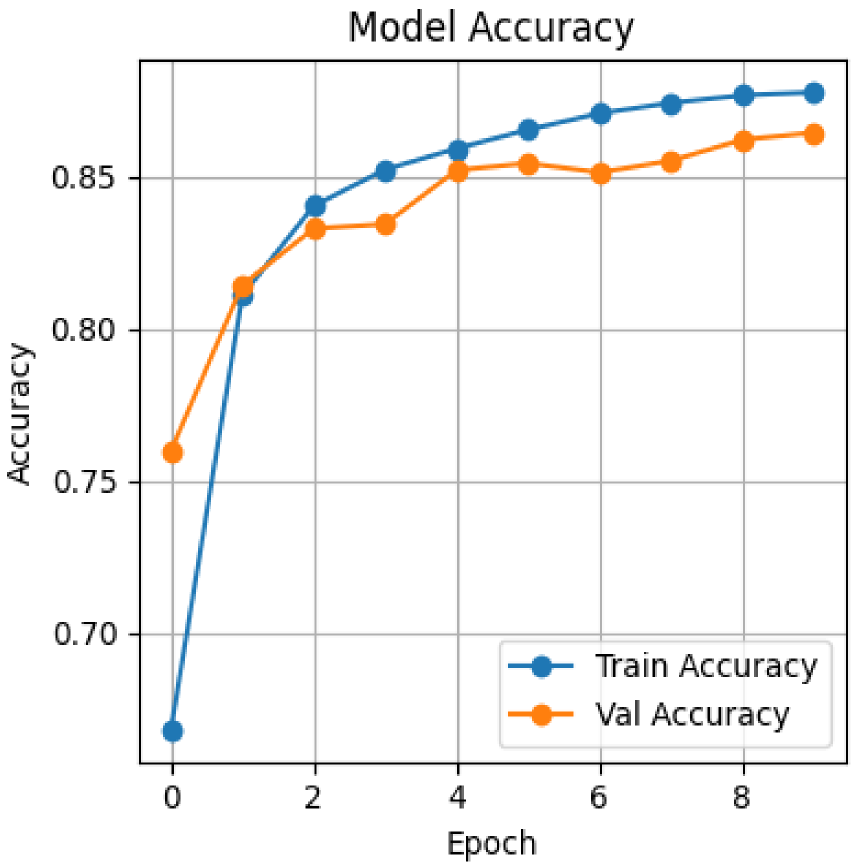
Why ResNet might be performing poorly?

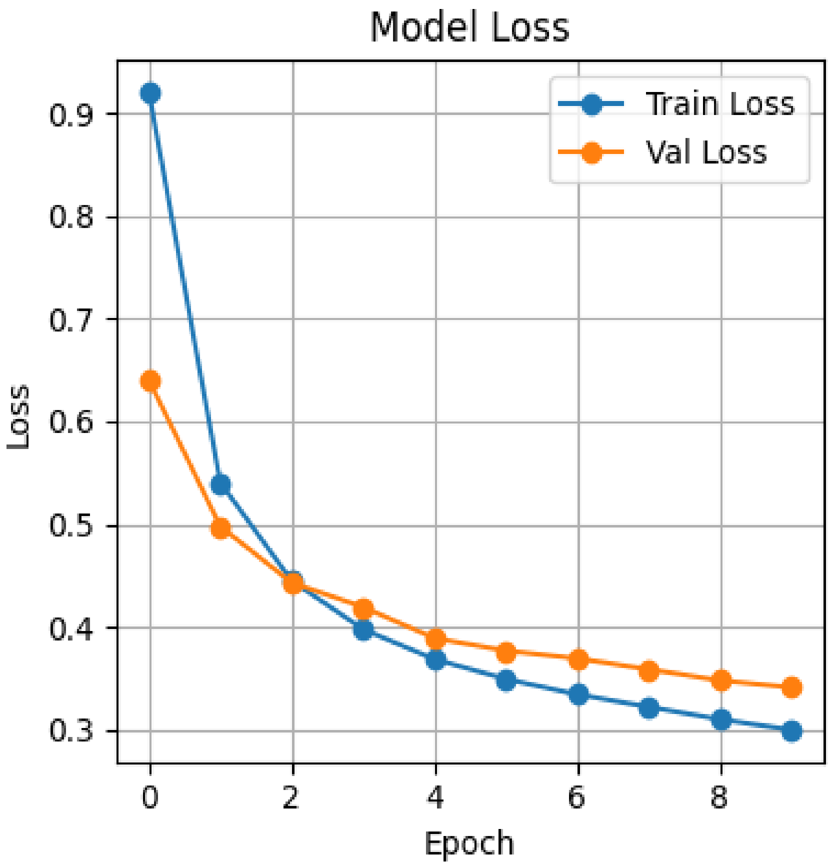
- Possible Reasons –
  - The early layers of ResNet (especially ResNet50/101) have **large initial kernels and stride**, which may lose subtle patterns in high-resolution CWT images early on.
  - CWT images are small (i.e, 128×128), deeper models like ResNet50 might not perform optimally due to aggressive downsampling in early layers.

Xception (transfer learning)

Transfer Learning Model Architecture

1. **Base Model:**
    1. Pretrained Xception
    2. Weights: *ImageNet*
    3. Top layers removed (include_top=False)
    4. Input shape: *(128, 128, 3)*
    5. All layers frozen
2. **Custom Head:**
    1. Flatten
    2. Dense (64 units, ReLU)
    3. Output: Dense (5 units, Softmax)
3. **Compilation:**
    1. Optimizer: Adam (learning rate = 1e-4)
    2. Loss: Categorical Crossentropy
    3. Metric: Accuracy



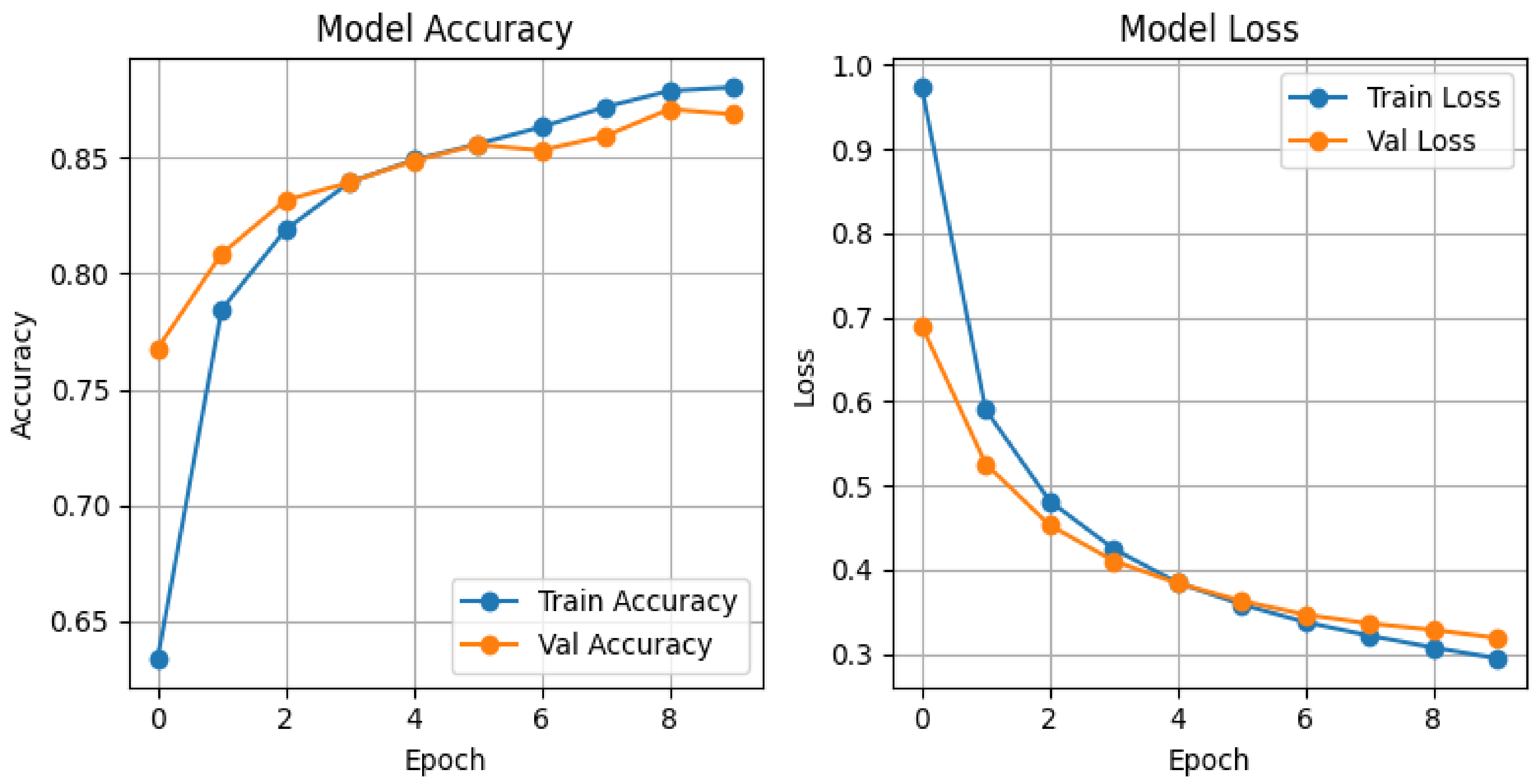


Validation Set Accuracy: 86.46%

DenseNet121 (transfer learning)

Transfer Learning Model Architecture

1. **Base Model:**
    1. Pretrained DenseNet121
    2. Weights: *ImageNet*
    3. Top layers removed (include_top=False)
    4. Input shape: *(128, 128, 3)*
    5. All layers frozen
2. **Custom Head:**
    1. Flatten
    2. Dense (64 units, ReLU)
    3. Output: Dense (5 units, Softmax)
3. **Compilation:**
    1. Optimizer: Adam (learning rate = 1e-4)
    2. Loss: Categorical Crossentropy
    3. Metric: Accuracy



Validation Set Accuracy: 86.85%