Comprehensive Model Report

Comparative Analysis of Deep Learning Models for EuroSAT Classification

This report provides a detailed comparison of three distinct Convolutional Neural Network (CNN) models used for classifying images from the EuroSAT dataset: a fine-tuned VGG16, a custom CNN with the Swish activation function, and a fine-tuned ResNet50.

---

1. VGG16 with Fine-Tuning

Model Architecture and Preprocessing

- **Base Model**: The VGG16 architecture, pre-trained on ImageNet, was used as the feature extraction base. The top classification layer of the original model was excluded (`include_top=False`).
- **Custom Head**: A new classification head was added on top of the base model. It consists of a `Flatten` layer, a `Dense` layer with 256 neurons and ReLU activation, a `Dropout` layer with a rate of 0.5 for regularization, and a final `Dense` output layer with softmax activation for the number of classes in the dataset.
- **Input Shape**: The model was configured to accept input images of size `64x64x3` .
- **Preprocessing**: Input images were resized to `64x64` and their pixel values were normalized by casting to `tf.float32` and scaling to a `[0.0, 1.0]` range.

Training Strategy

- **Two-Phase Training**:
  - *Initial Training*: The model was first trained for 10 epochs with the entire VGG16 base frozen (`base_model.trainable = False`).
  - *Fine-Tuning*: The VGG16 base was made trainable (`base_model.trainable = True`), but only the top 4 layers were unfrozen (`for layer in base_model.layers[:-4]: layer.trainable = False`).
- **Optimizer**: Adam (`1e-4` for initial training, `1e-5` for fine-tuning)
- **Loss Function**: `sparse_categorical_crossentropy`
- **Batch Size**: 32
- **Epochs**: Up to 30 (early stopping applied)
- **Callbacks**:
  - `EarlyStopping(patience=5)`
  - `ReduceLROnPlateau(patience=3)`

Performance

- **Initial Validation Accuracy**: 86.76%
- **Fine-Tuned Validation Accuracy**: 93.04%

---

2. Custom CNN with Swish Activation

Model Architecture and Preprocessing

- **Architecture**: A sequential model built from scratch, composed of three convolutional blocks followed by a dense classifier.
- **Conv Blocks**: Each block includes two `Conv2D` layers with Swish activation, `BatchNormalization` , `MaxPooling2D` , and `Dropout` (rate 0.3). Filter sizes increased across blocks: 32 → 64 → 128.
- **Classifier Head**: `Flatten` → `Dense(256, Swish)` → `BatchNormalization` → `Dropout(0.5)` → `Dense` softmax output
- **Input Shape**: `128x128x3`
- **Data Augmentation**: `RandomFlip` , `RandomRotation` , and `RandomZoom`
- **Preprocessing**: Images resized to `128x128` , scaled to `[0.0, 1.0]`

Training Strategy

- **Optimizer**: RMSprop (`learning_rate = 0.001`)
- **Loss Function**: `sparse_categorical_crossentropy`
- **Batch Size**: 32
- **Epochs**: Up to 50
- **Callbacks**:
  - `EarlyStopping(patience=7, monitor='val_accuracy')`
  - `ReduceLROnPlateau(patience=3)`
  - `ModelCheckpoint`

Performance

- **Peak Validation Accuracy**: 95.56%

---

3. ResNet50 with Fine-Tuning

Model Architecture and Preprocessing

- **Base Model**: Pre-trained ResNet50 (`include_top=False`)
- **Custom Head**: `GlobalAveragePooling2D` → `Dense(256, ReLU)` → `Dropout(0.5)` → softmax output
- **Input Shape**: `224x224x3`
- **Preprocessing**: Used `tf.keras.applications.resnet.preprocess_input` ; labels were one-hot encoded

Training Strategy

- **Two-Phase Training**:
  - *Initial Training*: 15 epochs with frozen base
  - *Fine-Tuning*: Top 10 layers unfrozen (`for layer in base_model.layers[:-10]: layer.trainable = False`), trained for 10 more epochs
- **Optimizer**: Adam (`1e-4` then `1e-5`)
- **Loss Function**: `categorical_crossentropy`
- **Batch Size**: 32
- **Callback**: `ModelCheckpoint(monitor='val_accuracy')`

Performance

- **Initial Validation Accuracy**: 96.96%
- **Fine-Tuned Validation Accuracy**: 97.04%

Model Comparison Table

Feature	VGG16 with Fine-Tuning	Custom CNN (with Swish)	ResNet50 with Fine-Tuning
Base Architecture	Pre-trained VGG16	Custom Sequential Model	Pre-trained ResNet50
Input Shape	`64x64x3`	`128x128x3`	`224x224x3`
Data Preprocessing	Resize, scale to `[0, 1]`	Resize, scale to `[0, 1]` , data augmentation	Resize, `resnet.preprocess_input`
Optimizer	Adam (LR: `1e-4` → `1e-5`)	RMSprop (LR: `0.001`)	Adam (LR: `1e-4` → `1e-5`)
Loss Function	`sparse_categorical_crossentropy`	`sparse_categorical_crossentropy`	`categorical_crossentropy`
Training Strategy	Fine-tuned top 4 layers after initial training	Trained from scratch	Fine-tuned top 10 layers after initial training
Peak Validation Accuracy	93.04%	95.56%	**97.04%**