Musical Instrument Detection Using Deep Learning

By: Nouran Hassan Ahmed 2022/00062

Course: Advanced AI

Presented to: Dr. Alaa Hamdy

Date: May 2025

Contents

Abstract

This project focuses on classifying musical instruments using deep learning techniques, utilizing a dataset of 30 classes. Multiple pre-trained CNN models (MobileNet, GoogLeNet, ResNet, VGG) were evaluated. With image augmentation and fine-tuning, MobileNet achieved 98% test accuracy.

---

1. Introduction

Musical instrument detection has applications in music education, automated tagging, and sound processing. This project explores image-based classification using convolutional neural networks (CNNs) including GoogleNet, MobileNet, ResNet, and VGG to identify 30 different musical instrument categories.

---

2. Dataset

- **Source:** Kaggle - gpiosenka/musical-instruments-image-classification
- **Classes:** 30 musical instruments
- **Format:** JPG, 224x224 pixels
- **Distribution:**
  - ~4793 training images
  - 150 validation images
  - 150 test images
- **Note:** Class imbalance was addressed through augmentation and class weighting.

---

3. Methodology

3.1 Data Preprocessing

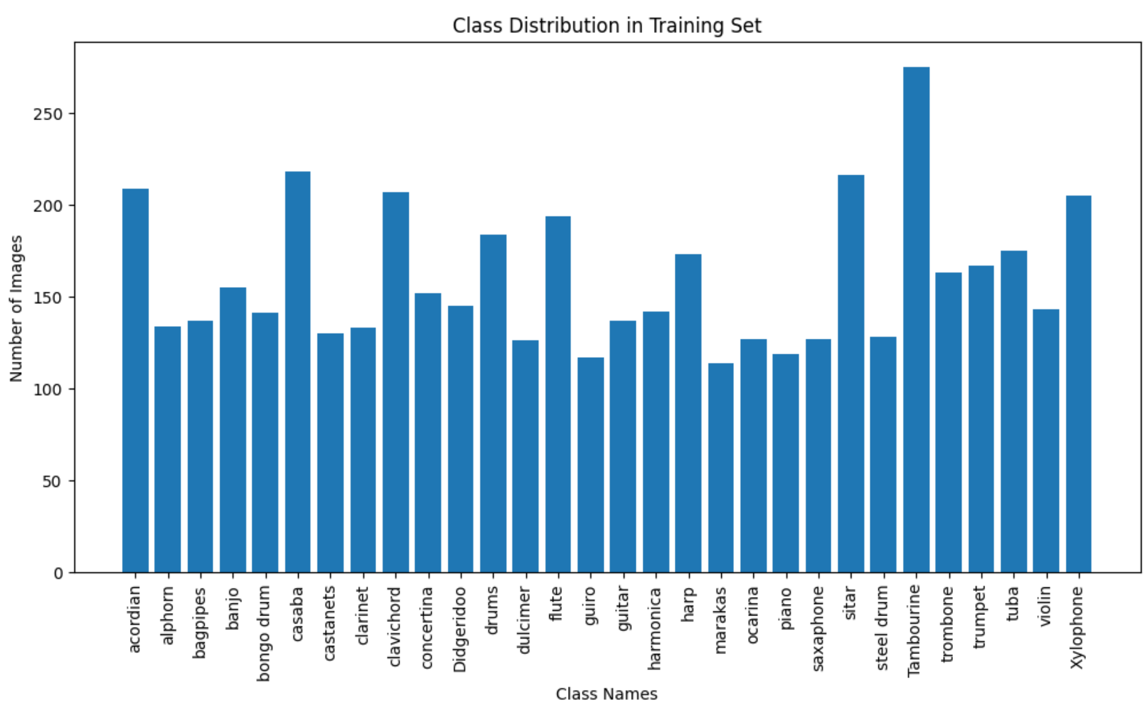
- Resizing: 224×224 pixels
- Normalization: [0,1] pixel values

- Data Augmentation:
  - Random rotations (±20°)
  - Width/height shifts (±20%)
  - Shearing, zooming, horizontal flipping

3.2 Handling Class Imbalance

Given that some instrument classes were underrepresented in the dataset, special attention was given to this issue during model training. A **dual approach** was employed to address class imbalance:

1. **Data Augmentation:**
   As described above, aggressive augmentation was applied to the training set to artificially boost underrepresented classes and reduce overfitting.
2. **Class Weighting:**
   In addition to augmentation, **class weights were computed** using `sklearn.utils.class_weight.compute_class_weight` from the `scikit-learn` library. These weights ensure that underrepresented classes contribute more significantly to the loss function during training.



3.3 Models Used

Pre-trained CNNs with transfer learning:

- MobileNet
- GoogLeNet
- ResNet50
- VGG16

Model customization:

- `include_top=False`
- GlobalAveragePooling2D
- Dense layers
- Dropout (0.5)
- Softmax output

3.3 Training Process

- Phase 1: Train with frozen base layers
- Phase 2: Fine-tune top 100 layers
- **Hyperparameters:**
    - Learning rate: 0.0001
    - Batch size: 64
    - Callbacks: EarlyStopping (patience=10), ReduceLROnPlateau

---

4. Results & Evaluation

4.1 MobileNet Performance

- Accuracy: 98%
- Loss: 0.0564
- Best generalization on unseen images

4.2 Model Comparison Table

Model	Accuracy	Training Time	Parameters	Notes
MobileNet	98%	~120 mins	4.2M	Lightweight, fast, efficient
InceptionV3	98%	~180 mins	23.8M	Parallel convolutions
ResNet50	94.67%	~150 mins	25.5M	Deep residual architecture

Model	Accuracy	Training Time	Parameters	Notes
VGG16	89.33%	~210 mins	138M	Simple but memory intensive

5. Model Insights

5.1 Key Architecture Features

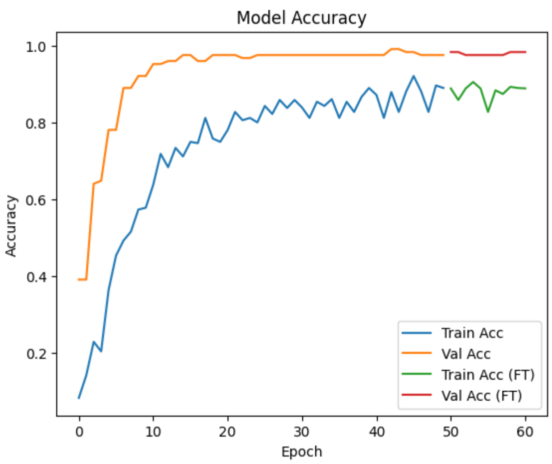
```
# MobileNet
x = GlobalAveragePooling2D()(x)
x = Dense(512)(x)
x = Dropout(0.5)(x)

# InceptionV3
x = GlobalAveragePooling2D()(x)
x = Dense(1024)(x)
x = Dropout(0.5)(x)

# ResNet50
x = GlobalAveragePooling2D()(x)
x = Dense(512)(x)
x = BatchNormalization()(x)

# VGG16
x = GlobalAveragePooling2D()(x)
x = Dense(1024)(x)
x = Dropout(0.5)(x)
```

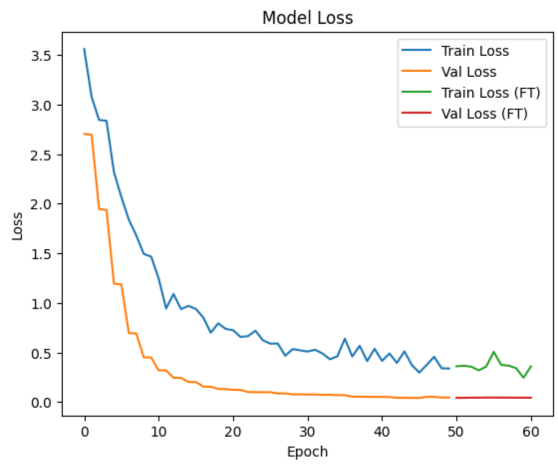
5.2 Observations

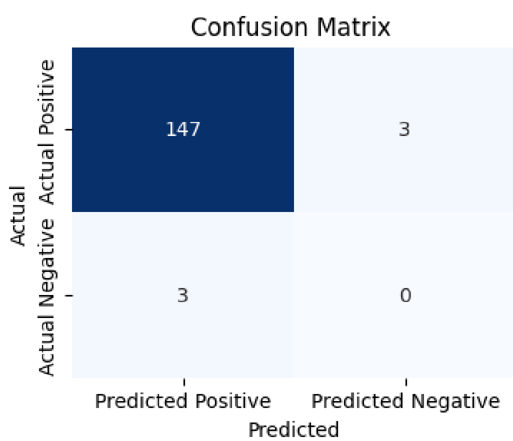
- MobileNet and InceptionV3 tied at 98% accuracy.
- MobileNet trained 1.5× faster than InceptionV3.
- MobileNet had the most stable validation curve.
- VGG16 used 3× more memory than MobileNet.

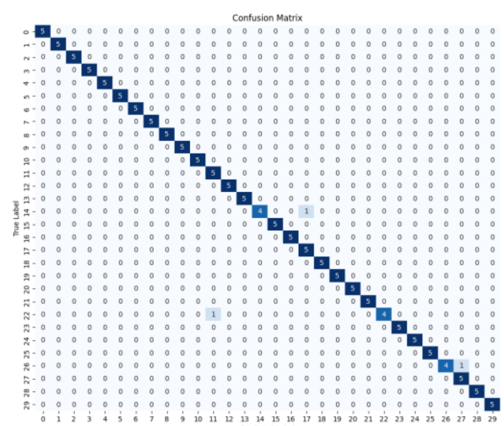
6. Conclusion

- MobileNet achieved the best overall performance.
- Data augmentation mitigated class imbalance.
- Transfer learning was highly effective.
- Potential applications: music education tools, inventory systems, real-time classification apps.









7. Future Work

- Integrate audio-based detection
- Experiment with ensemble methods
- Deploy as a web or mobile application

8. Recommendations

Use MobileNet for:

- Mobile apps

- Real-time detection

- Low-resource environments

Alternatives:

- InceptionV3: Best for server-side applications

- ResNet50: Best for complex visual patterns

- VGG16: Avoid due to inefficiency

Possible Improvements:

1. Ensemble MobileNet + InceptionV3

2. Add attention mechanisms

3. Use test-time augmentation

9. Saved Model

- File: `musical_instruments_model_Mobile_Net.h5`

---

10. References

- Kaggle: gpiosenka/musical-instruments-image-classification
- TensorFlow & Keras Documentation
- CNN & Transfer Learning Research Papers