**Scene Classification Model Report:**

Built a **robust indoor/outdoor classifier** for our **SafeWalk AI-powered navigation system** — optimized for accuracy, efficiency, and real-world usability.

---

**Summary of All Experiments & Outcomes**

| Version | Model/Approach | Dataset(s) Used | Outcome | Issues Faced | Result |
|---|---|---|---|---|---|
| **V1** | MobileNetV2 (TF/Keras) | MIT Indoor + SUN Outdoor | ~90% acc | Crashed due to OOM on Kaggle + dataset and evaluation bias + overfitting for one class+ label ambiguity and class imbalance etc. | Rejected |
| **V2** | Lightweight Custom CNN | MIT + CIFAR10 fallback | 85–90% acc | Same problems as (V1)+Overfit fast, lacked generalization+ resolution and task mismatch(MIT is for scene classification while CIFAR10 is for object classification) | Rejected |
| **V3** | EfficientNet-B0 (PyTorch) | MIT + SUN (symlinked) | 73.65% val acc | Limited accuracy, despite TTA improvements (~76%), worked properly on the dataset this time avoided problems in V1 but another problem was discovered in SUN and Places365(another quick version) that came in the form of label noise and semantic ambiguity where classes like | Rejected |

| | | | | | |
|---|---|---|---|---|---|
| | | | | candy shop and butcher shop are labeled as "indoor", while the actual **scene content** is sometimes **outdoor or semi-outdoor or even all three classes** | |
| **V4** | EfficientNetB0 (TF/Keras) | MIT Indoor + LabelMe | ~93% acc | Some class imbalance, model was solid on test set but labeled all new images as indoor clearly overfitting | Rejected |
| **V5** | MobileNetV3Large (TF/Keras) | ADE20K (manually relabeled) | **82.24% val acc** | .h5 loading failed; model overfit after phase 3 | Rejected |
| **V6** | FINAL: MobileNetV3Large (TF/Keras) | ADE20K(manually relabeled/added extra classes) | **80.13% val acc** | Slow training; overfit prevented by staged training + reg+ increased the number of classes for classifying the image as indoor or outdoor | ✅ **Final Pick** |

---

**What Went Wrong (Lessons from Experiments)**

**Version Mismatch + Loading Issues**

- **.h5 format** caused major compatibility errors in Keras 3.x and TF 2.x.
- *Solution:* Used modern .keras format with correct architecture reconstruction.

**Memory & Resource Constraints (Kaggle)**

- Early models like MobileNetV2 or large EfficientNet versions **crashed or were slow**.
- *Solution:* Downscaled input sizes, used lighter models like MobileNetV3.

**Model Overfitting**

- Overfitting occurred even with dropout and augmentations in some variants.

- *Solution:* Introduced **staged unfreezing**, **OneCycleLR**, **label smoothing**, and **early stopping**.

**Low Initial Accuracy**

- Some models started with <60% validation accuracy.
- *Solution:* Improved with aggressive augmentations, class balancing, and learning rate scheduling.

---

**Final Chosen Model Summary**

**Model**

- **MobileNetV3Large (TF/Keras)** with staged fine-tuning
- Saved as .keras (no version issues)

**Training Strategy**

- **Phase 1**: Freeze base, train top layers(feature extraction)
- **Phase 2**: Unfreeze last 50 layers, reduce LR(adapts deeper layers to our domain.)
- **Phase 3**: Fine-tune full model (few epochs, low LR)(maximizes performance)

🧰 **Key Optimizations**

- Label smoothing (0.1), dropout (0.5), L2 regularization
- EarlyStopping + ReduceLROnPlateau
- Strong augmentation (rotation, zoom, flip, brightness)
- Balanced dataset with ~10000 images/class(10000 indoor and ~9900+ outdoor to be more precise)

📈 **Final Results**

| Metric | Value |
|---|---|
| Accuracy | **80.13%** |
| Loss | 0.39 |
| Inference | ~50ms/image |

| Metric | Value |
|---|---|
| Format | .keras |

🧠 **Lessons Learned**

- Start lightweight and scale up
- Don’t trust .h5 in modern workflows
- Gradual unfreezing outperforms immediate fine-tuning
- Test-Time Augmentation (TTA) is a low-cost boost

**Final model workflow**

**1. Imports and Setup**

- Imported essential libraries: os, numpy, pandas, PIL, sklearn, tensorflow, and tensorflow_hub.
- Set up dataset paths for ADE20K images and masks.

**2. Class Mapping and Labeling**

- Loaded and cleaned class mappings from objectInfo150.txt.
- Defined lists of indoor and outdoor object classes.
- Mapped object names to their corresponding IDs for labeling.

**3. Label Generation**

- Implemented a function to assign each image a label (0 for indoor, 1 for outdoor) based on the pixel coverage of class IDs in the mask.
- Processed all images, skipping those with ambiguous labels.
- Created a DataFrame with image paths and labels.
- Achieved a reasonably balanced dataset(10000 indoor(0), and around 9900+ ourdoor(1)), with label distribution printed for verification.
- Saved the labeled dataset to ade20k_labels.csv.

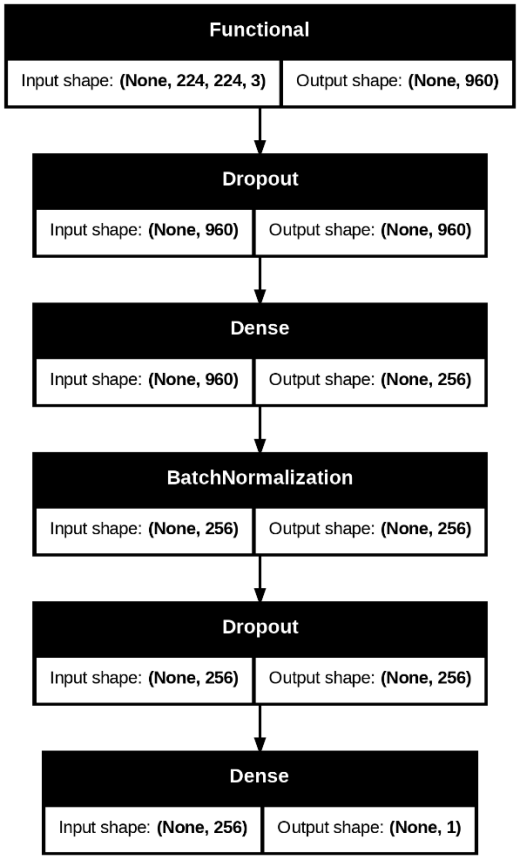
**4. Data Preparation**

- Split the labeled data into training and validation sets (80/20 split).

- Used ImageDataGenerator for data augmentation (rotation, shift, flip, etc.) and normalization.
- Created data generators for both training and validation sets.

**5. Model Architecture and Training**

- Used MobileNetV3Large (pretrained on ImageNet) as the base model.
- Added dropout, dense, and batch normalization layers for regularization.
- Training proceeded in three phases:
    - **Phase 1:** Feature extraction (base model frozen).
    - **Phase 2:** Partial fine-tuning (unfroze top layers).
    - **Phase 3:** Full fine-tuning (unfroze more layers if validation accuracy > 0.7).
- Used callbacks: EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint.
- Achieved best validation accuracy by restoring the best model weights




**Total params:** 7,453,381 (28.43 MB)

**Trainable params:** 2,104,865 (8.03 MB)

**Non-trainable params:** 1,138,784 (4.34 MB)

**Optimizer params:** 4,209,732 (16.06 MB)

***Parameter statistics of the model***

***Total params:*** *Total number of* ***weights*** *and* ***biases*** *in our model*

***Trainable params:*** *The parameters to be updated during back prop.*

***Non-trainable params:*** *Frozen params part of the model but will not change during training*

***Optimizer params:*** *Params used internally by the optimizer(Adam for our model)*

*Figure 1-Keras_Model_Architecture_Diagram*

### 6. Evaluation and Export

- Evaluated the final model on the validation set.
- Saved the model in both .keras and quantized .tflite formats for deployment.

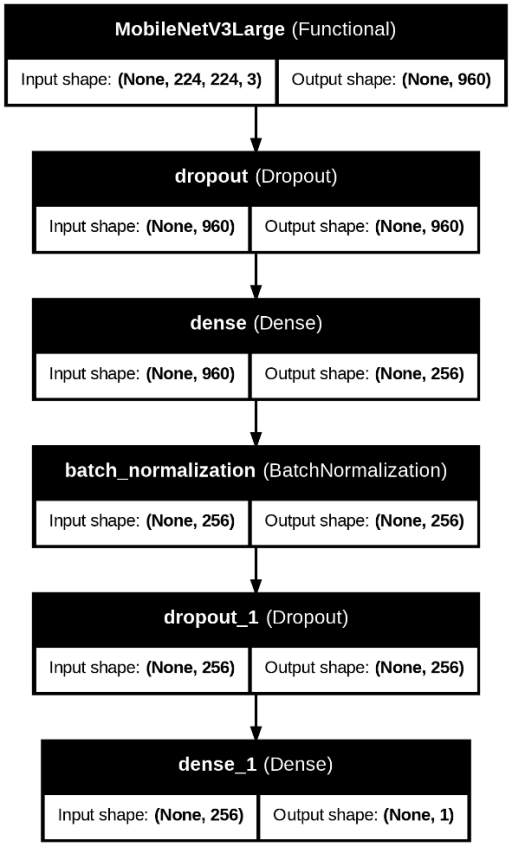



*Figure 2-Model_Pipline/Workflow*

---

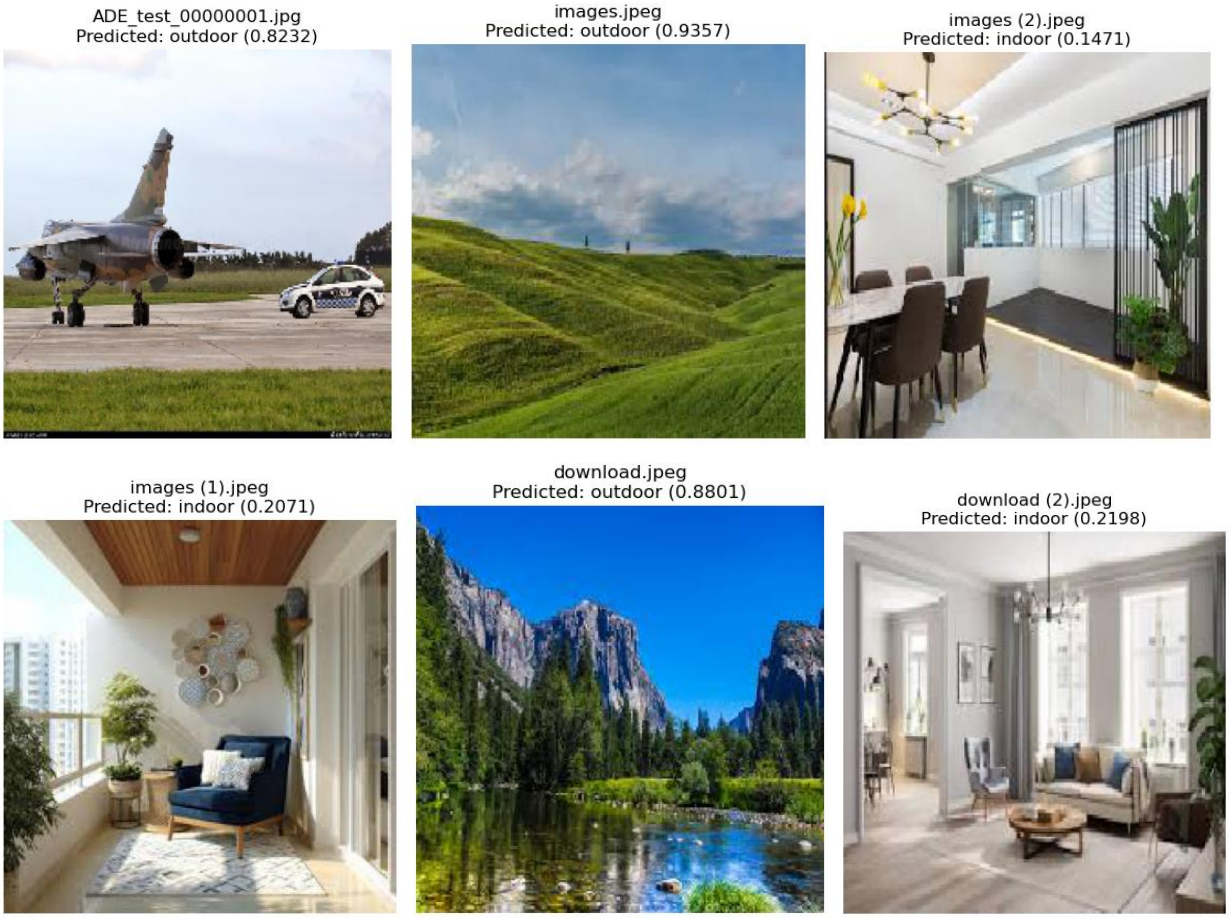
### Key Outcomes

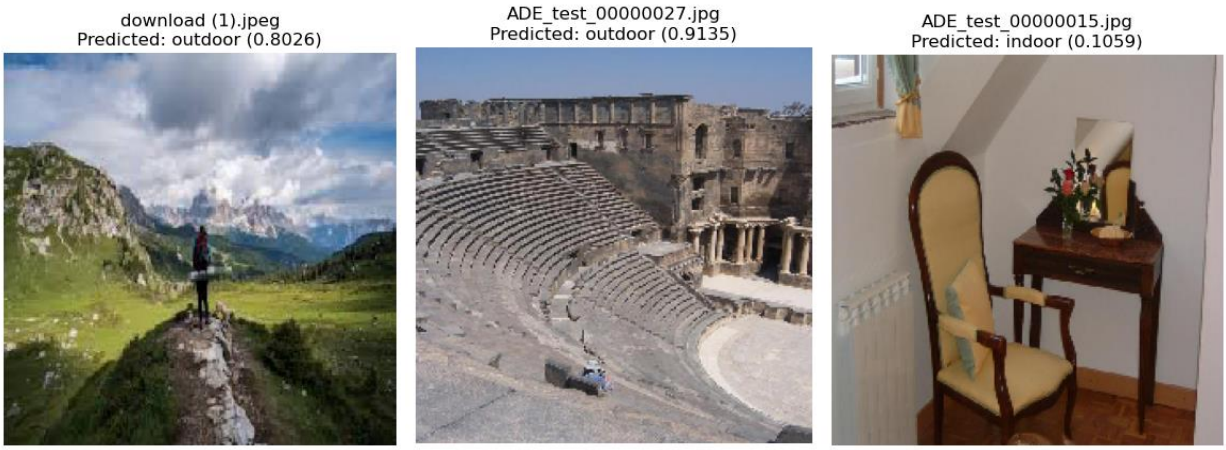
- Successfully labeled and split the ADE20K dataset for indoor/outdoor scene classification.
- Built and trained a robust deep learning model with transfer learning and strong regularization.
- Achieved balanced label distribution and good validation performance (see notebook for exact metrics).
- Exported the model for both standard and mobile/edge deployment.

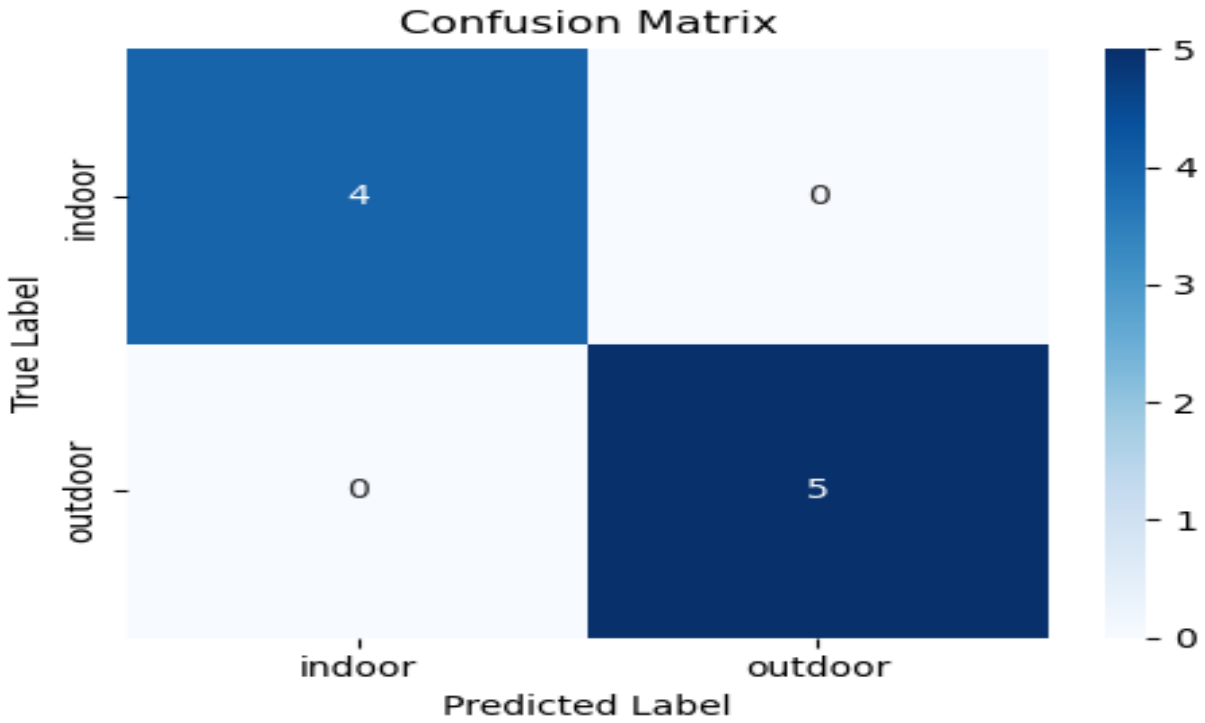
---

**Test Results(Visualizations)**

The model was given 9 random unseen images downloaded from Google the images contained both indoor and outdoor scenes and the model was able to identify all photos correctly regardless of the different resolutions and shapes of the images there was also a photo of a selfie from a friend that was identified as indoor correctly(the photo is not included in the report you just have to take our word for it). the results are present below:

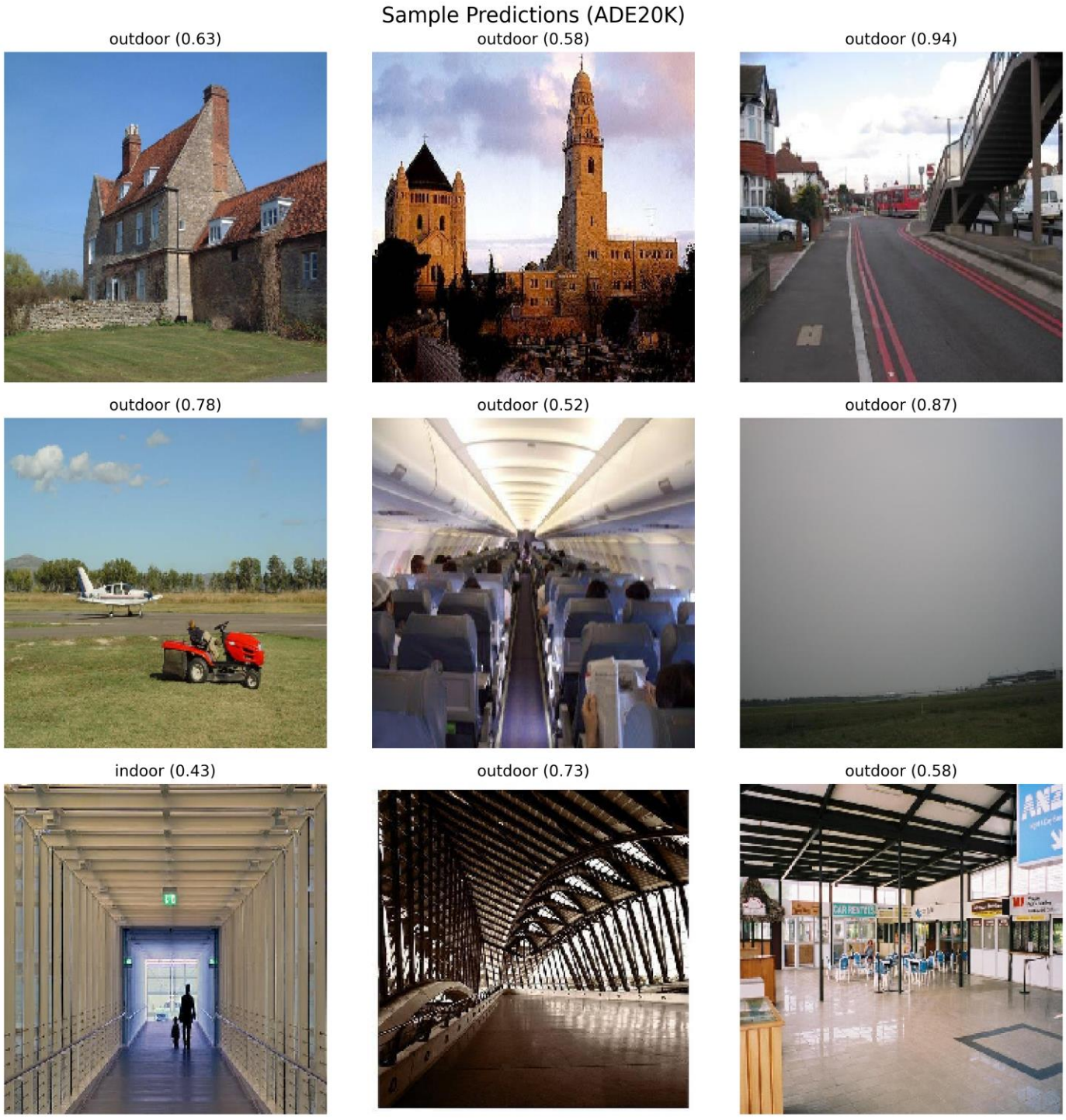


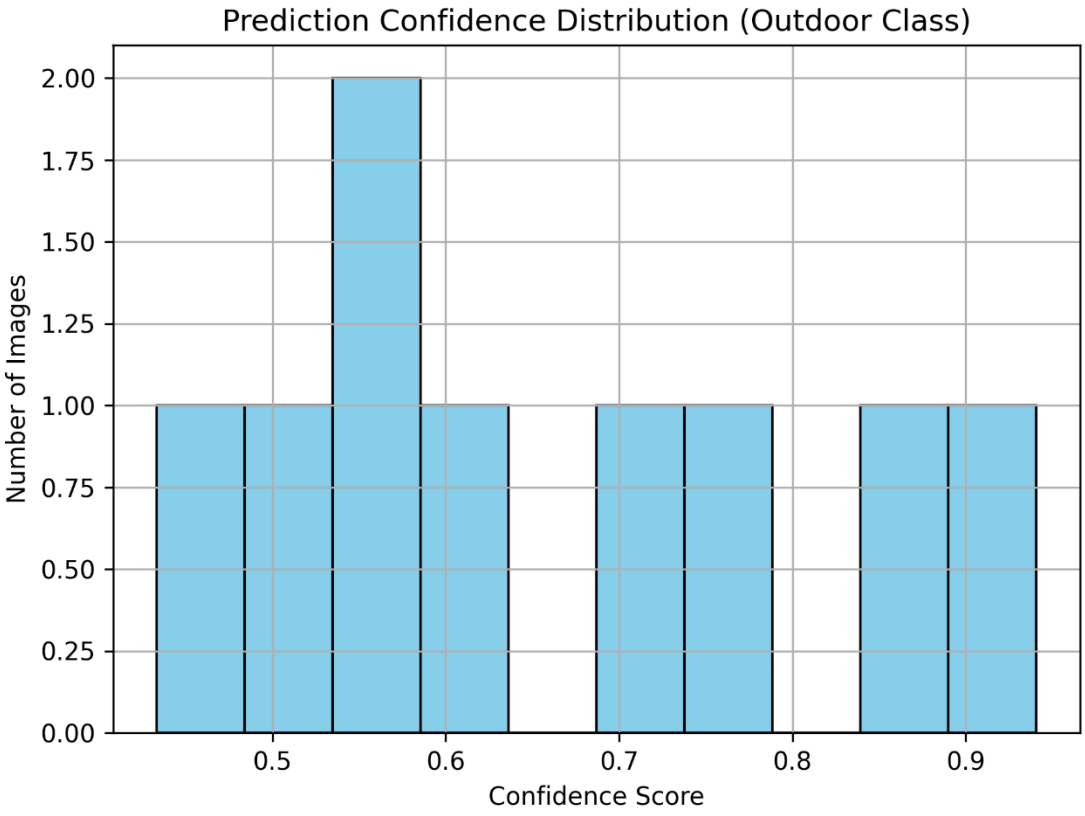



**Test Set visualizations:** The results the model gave on images from the test split from the ADE20K used to train the model the images show that most images where classified correctly but with some misses also an image of the confidence distribution for the outdoor images is

presented in the images below.






---

**Summary:**

This paper demonstrates all approaches to get the best model and a full pipeline for indoor/outdoor scene classification using ADE20K(the model that was chosen at the end), including data labeling, augmentation, transfer learning, evaluation, and export for deployment. The report includes photos that will help readers visualize model performance for the models real world use performance.