WASTE To WOW

Home Waste Management Recommendation System - Recycling or Upcycling

Team 7

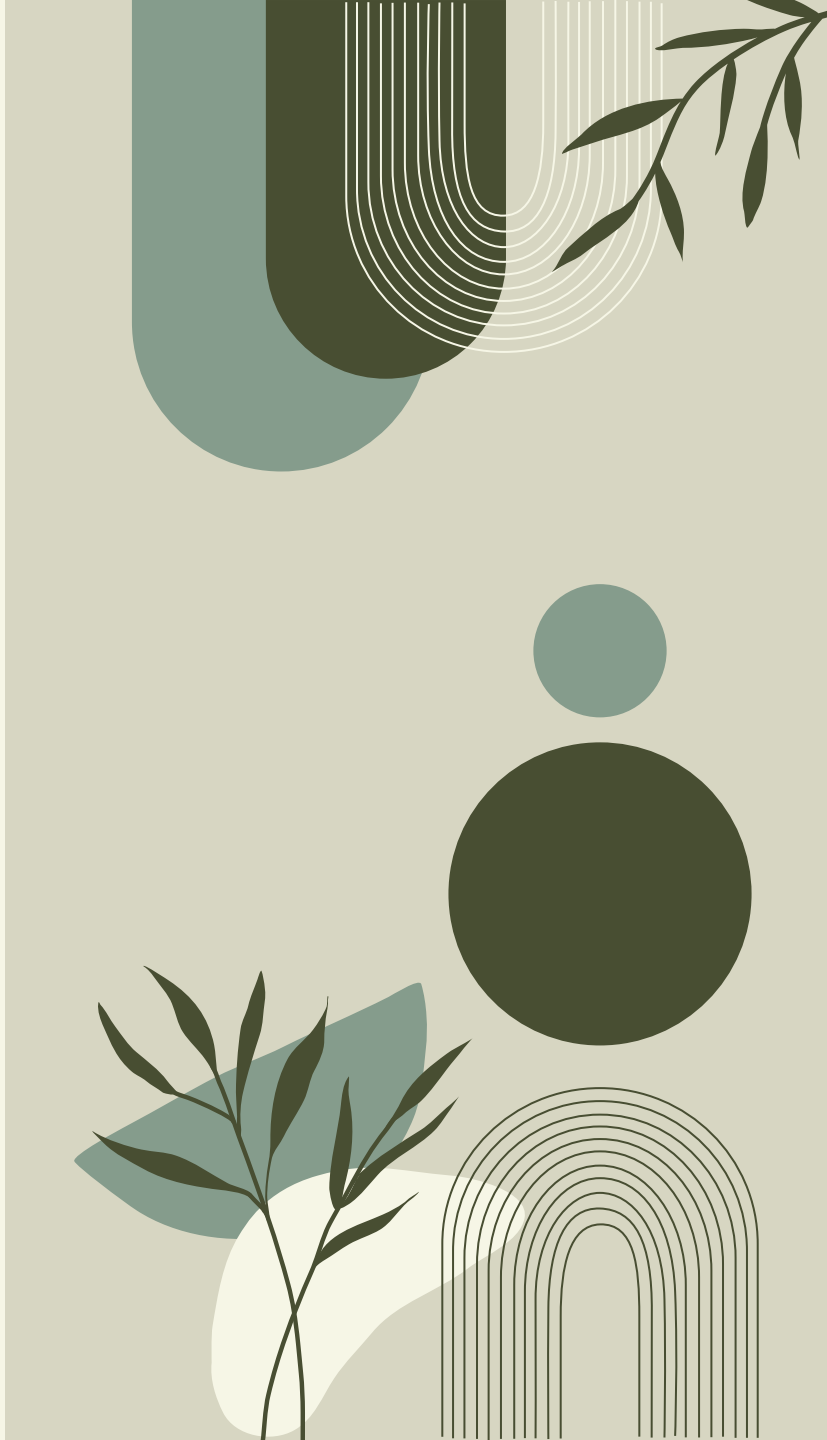


Table of contents.

ResNet50

Model Proposal

- Depth in CNNs: ResNet-50 tackles the vanishing/exploding gradient problem by facilitating the training of much deeper networks.
- Innovative Framework: Introduces residual learning which focuses on learning residual functions, significantly different from traditional deep learning approaches.
- Residual Building Blocks: Utilizes shortcut connections that perform identity mapping and simplify the learning process, as depicted in the model's architecture figures.
- Structured Design: Follows a consistent filter strategy that aligns with the network's depth and complexity, ensuring efficient feature mapping.

Model Proposal

- Dimensionality Handling: Employs projection shortcuts for dimension matching in the layers, optimizing the flow of information through the network.
- Overcoming Degradation: Effectively addresses the degradation problem, allowing deeper networks to maintain or improve performance over shallower counterparts.

Model Support

Hardware Configuration

Component	Specification
CPU	9th Gen Intel(R) Core(TM) i5-9300H @ 2.40GHz
GPU	Nvidia GeForce GTX 1650 4 GB
RAM	16.0 DDR4 GB (15.8 GB usable)

Software Configuration

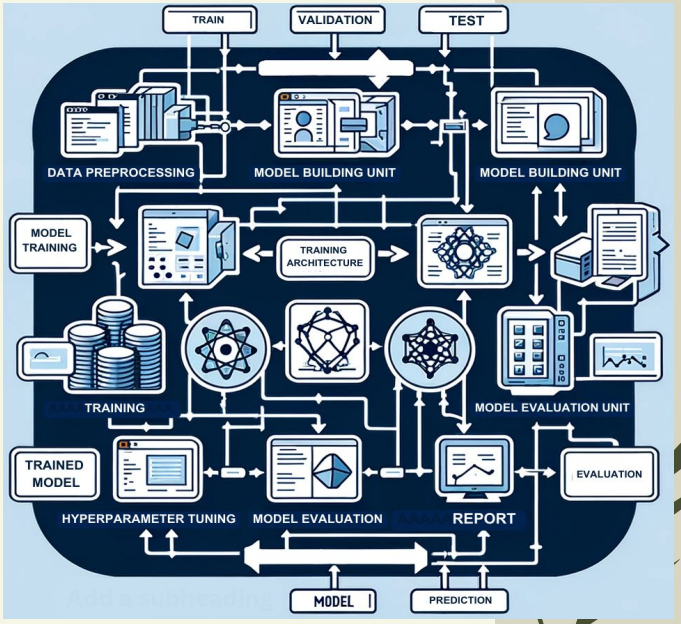
Component	Specification
OS	Windows 11 Home
IDE	Jupyter Notebook

Tools

Library/Tool	Purpose	Functions/Methods Used
TensorFlow	Primary deep learning framework	keras.applications.ResNet50, keras.layers.GlobalAveragePooling2D, keras.layers.Dense, keras.layers.Dropout, keras.Model, keras.callbacks.EarlyStopping, keras.callbacks.ModelCheckpoint, keras.callbacks.ReduceLROnPlateau
Keras Tuner	For optimizing the hyperparameters of neural networks	RandomSearch
NumPy	Array processing for numbers, strings, records, etc.	np.random.seed
Matplotlib	Plotting library for Python and its numerical extensions	plt.plot, plt.title, plt.ylabel, plt.xlabel, plt.legend, plt.show
Scikit-learn	Machine learning library for Python	classification_report, confusion_matrix

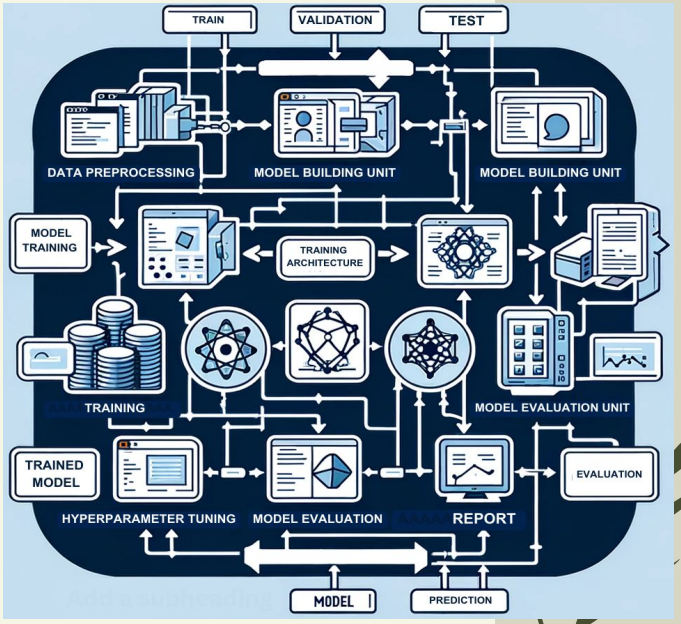
Model Data Flow Diagram

- Data Preprocessing: Utilizes TensorFlow's ImageDataGenerator for image resizing, normalization, and augmentation to enhance model generalizability and prevent overfitting.
- Model Building: Adopts ResNet50 architecture with custom layers (GlobalAveragePooling2D, Dense, Dropout) for classifying images into 11 classes, implemented within TensorFlow's Model class.
- Training and Hyperparameter Tuning: Employs iterative learning with the train_generator object and uses Keras Tuner's RandomSearch for optimizing key hyperparameters like neuron count in Dense layers and learning rate.



Model Data Flow Diagram

- Model Evaluation and Reporting: Evaluates performance on unseen datasets, providing metrics (loss, accuracy), and detailed classification reports including precision, recall, f1-scores, and confusion matrices.
- Model Prediction and Persistence: Tests generalization on new data and saves the trained model for future use without the need for retraining, ensuring practical deployment readiness.



Model Evaluation Results

Biowaste	[[218 0 0 1 1 2 1 3 1 3 1]
Cardboard	[6 147 0 2 1 0 1 2 11 0 0]
Clothes	[0 0 248 1 0 0 0 0 1 0 0]
Ewaste	[0 1 0 198 0 1 4 7 1 1 1]
Furniture	[0 0 0 0 53 0 0 0 0 0 0]
Glass	[1 4 0 0 0 146 4 3 2 9 0]
Medical	[3 2 1 3 0 1 115 4 1 9 0]
Metal	[4 3 0 5 0 3 0 170 4 2 1]
Paper	[2 8 0 2 0 1 3 2 167 11 1]
Plastic	[2 0 1 3 0 7 4 3 9 160 0]
Shoes	[2 1 1 0 0 0 0 1 0 1 191]]

Model Evaluation Results

- High Precision and Recall: The model exhibits high precision and recall for most categories, indicating effective learning and accurate predictions.
- Exceptional Performance for 'Clothes' and 'Shoes': Achieves the highest precision and recall specifically in the 'clothes' and 'shoes' categories, showcasing the model's strong ability to differentiate these items.
- Lower Performance for 'Plastic': Exhibits lower precision and recall in the 'plastic' category, suggesting a need for additional training data or a reevaluation of its feature representations.
- Overall Robustness: Maintains an overall accuracy of 0.91 and a weighted average f1-score of 0.91, demonstrating strong and consistent performance across various classes.

	precision	recall	f1-score	support
biowaste	0.92	0.94	0.93	231
cardboard	0.89	0.86	0.88	170
clothes	0.99	0.99	0.99	250
e-waste	0.92	0.93	0.92	214
furniture	0.96	1.00	0.98	53
glass	0.91	0.86	0.88	169
medical	0.87	0.83	0.85	139
metal	0.87	0.89	0.88	192
paper	0.85	0.85	0.85	197
plastic	0.82	0.85	0.83	189
shoes	0.98	0.97	0.97	197
accuracy			0.91	2001
macro avg	0.91	0.91	0.91	2001
weighted avg	0.91	0.91	0.91	2001

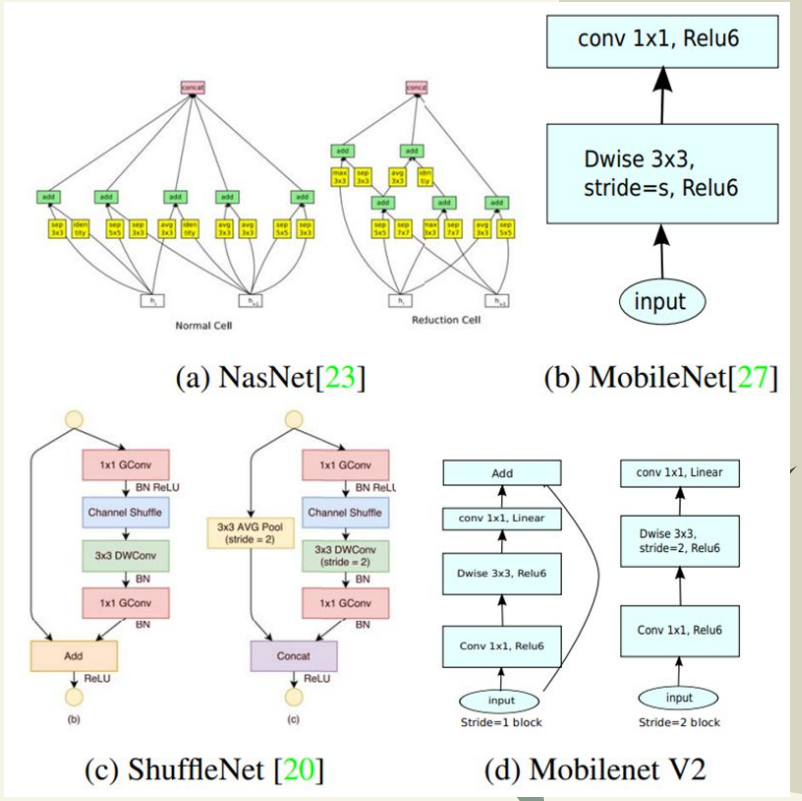
MobileNetV2

Model Proposal

- MobileNetV2 is a neural network architecture optimized for mobile and embedded devices, balancing efficiency and accuracy.
- Innovations include the inverted residual with a linear bottleneck for processing compressed representations effectively.
- It builds on prior architectures by incorporating depthwise separable convolutions, linear bottlenecks, and inverted residuals, following advances in deep learning optimization techniques.
- Depthwise separable convolutions reduce computational costs by 8 to 9 times compared to standard convolutions.
- Linear bottlenecks capture low-dimensional manifolds in neural network activations, maintaining information integrity while reducing dimensionality.

Model Proposal

- Inverted residual structures connect shortcuts between bottlenecks, improving gradient flow and memory efficiency.
- The architecture consists of an initial convolution layer followed by 19 residual bottleneck layers, with scalability to adjust to various performance needs by modifying width multipliers and input resolution.
- MobileNetV2 minimizes memory access during inference, which is advantageous for devices with limited memory.
- The model has been proven effective across various applications, including image classification, object detection, and semantic segmentation, showing high accuracy with fewer parameters and reduced computational complexity.



Model Support

Hardware Configuration

Component	Specification
CPU	11th Gen Intel(R) Core(TM) i5-1155G7 @ 2.50GHz 2.50 GHz
GPU	Intel(R) Iris(R) Xe Graphics
RAM	16.0 GB (15.8 GB usable)

Software Configuration

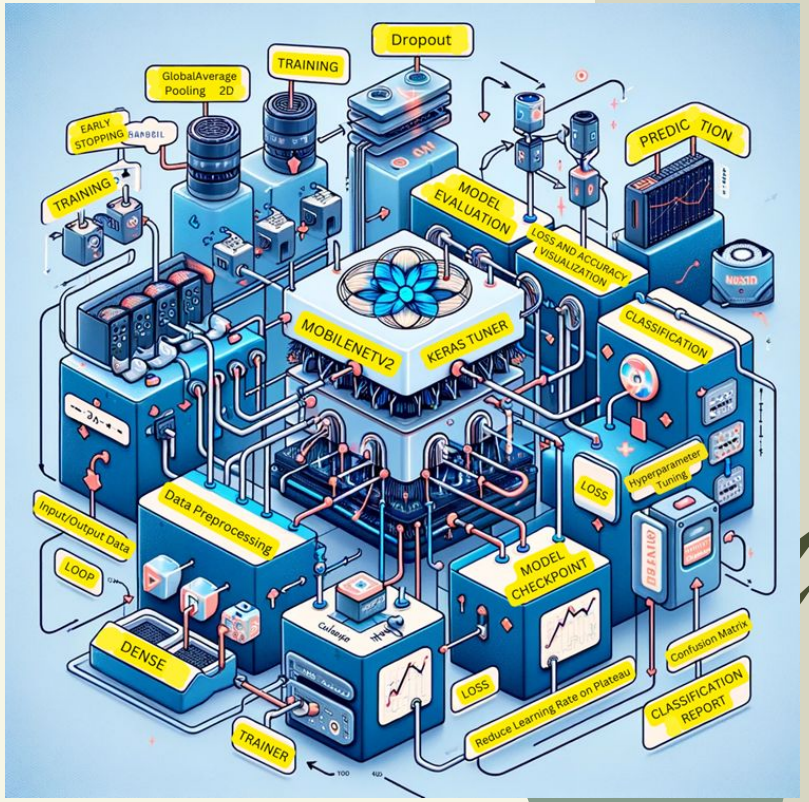
Component	Specification
OS	Edition - Windows 11 Home Version - 22H2
IDE	PyCharm 2023.2.4

Tools

Library/Tool	Purpose	Functions/Methods Used
TensorFlow	Primary deep learning framework	Keras API, ImageDataGenerator, MobileNetV2, GlobalAveragePooling2D, Dense, Dropout, Model, EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
Keras Tuner	For optimizing the hyperparameters of neural networks	RandomSearch
NumPy	Array processing for numbers, strings, records, etc.	Random seed setting
Matplotlib	Plotting library for Python and its numerical extensions	Plotting training & validation accuracy and loss
Scikit-learn	Machine learning library for Python	classification_report, confusion_matrix

Model Data Flow Diagram

- MobileNetV2 is the central architecture, chosen for its balance of efficiency and accuracy, using depthwise separable convolutions.
- Post-convolutional layers, the workflow includes a GlobalAveragePooling2D layer for dimensionality reduction and Dropout for regularization.
- Dense layers follow for the final classification computation, based on extracted features.
- Keras Tuner conducts hyperparameter optimization to find the best model parameters.
- Callbacks like EarlyStopping, ModelCheckpoint, and ReduceLROnPlateau regulate the training process to prevent overfitting and optimize learning.
- After training, model evaluation is performed using the test set to compute loss and accuracy.



Model Evaluation Results

Biowaste	[[210 0 0 3 2 3 1 6 1 3 2]
Cardboard	[6 139 1 5 0 3 4 4 7 0 1]
Clothes	[1 0 247 1 0 0 1 0 0 0 0]
Ewaste	[2 1 1 189 0 0 7 8 0 0 6]
Furniture	[0 0 0 0 53 0 0 0 0 0 0]
Glass	[2 3 0 0 0 145 7 6 1 4 1]
Medical	[7 1 1 4 0 2 115 2 3 3 1]
Metal	[3 4 1 3 0 5 3 168 0 4 1]
Paper	[9 17 1 3 0 3 9 7 137 8 3]
Plastic	[6 1 0 1 1 6 7 15 6 145 1]
Shoes	[2 0 2 0 1 1 0 0 0 0 191]]

Model Evaluation Results

- The model has an overall accuracy of 87%, correctly predicting class labels most of the time.
- Both macro and weighted averages for precision, recall, and F1-score are provided, reflecting consistent performance across all classes.
- Despite the challenges in classifying numerous waste material types, the model demonstrates robust performance.
- There is potential for improvement, suggesting that a different model architecture could yield better results.

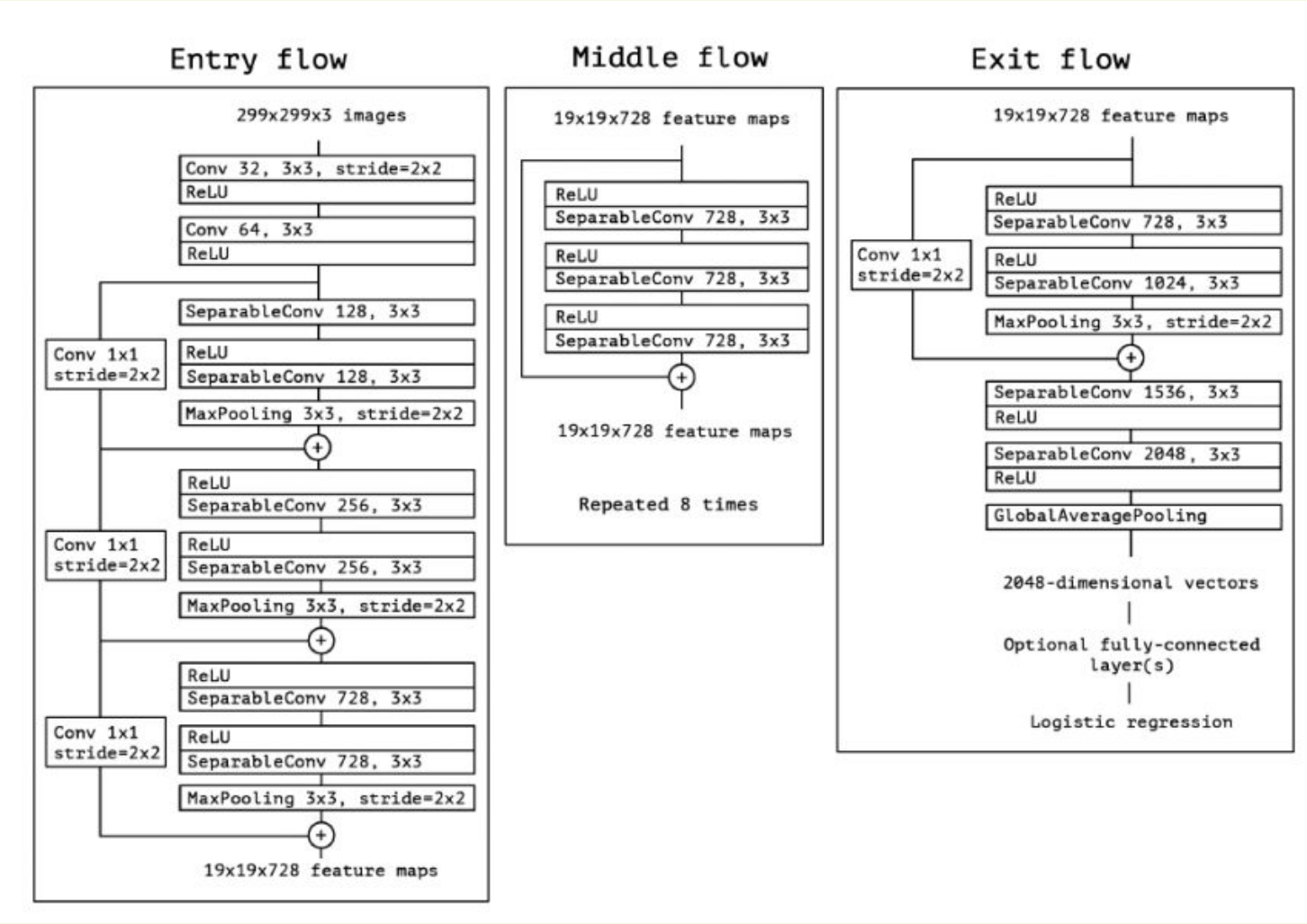
	precision	recall	f1-score	support
biowaste	0.85	0.91	0.88	231
cardboard	0.84	0.82	0.83	170
clothes	0.97	0.99	0.98	250
e-waste	0.90	0.88	0.89	214
furniture	0.93	1.00	0.96	53
glass	0.86	0.86	0.86	169
medical	0.75	0.83	0.78	139
metal	0.78	0.88	0.82	192
paper	0.88	0.70	0.78	197
plastic	0.87	0.77	0.81	189
shoes	0.92	0.97	0.95	197
accuracy			0.87	2001
macro avg	0.87	0.87	0.87	2001
weighted avg	0.87	0.87	0.87	2001

Xception

Model Proposal

- Xception is a state-of-the-art neural network architecture designed to excel in capturing intricate hierarchical features in images.
- Its unique approach involves entirely decoupling cross-channels and spatial correlations in feature maps, a hypothesis that goes beyond the inception module.
- Named "Extreme Inception," Xception represents a significant advancement in deep learning architectures.
- Hierarchical Feature Extraction: Each layer in the hierarchy focuses on different levels of abstraction, allowing the model to learn representations ranging from simple to complex features.
- Depthwise Separable Convolutions: Instead of using traditional convolutions, Xception employs depth wise separable convolutions. A depthwise separable convolution consists of two steps: depthwise convolutions and pointwise convolutions.

Model Architecture



Model Support

Hardware Configuration

Component	Specification
CPU	11th Gen Intel(R) Core(TM)
GPU	Intel(R) UHD Graphics 620
RAM	16.0 GB

Software Configuration

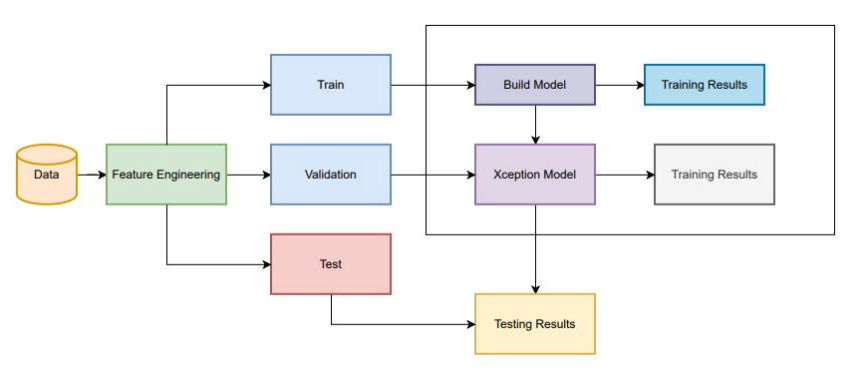
Component	Specification
OS	Windows 11
IDE	Google Colab

Tools

Library/Tool	Purpose	Functions/Methods Used
TensorFlow	Primary deep learning framework	Keras AP, Xception, GlobalAveragePooling2D, Dense
Matplotlib	Plotting library for Python and its numerical extensions	Plotting training & validation accuracy and loss
Scikit-learn	Machine learning library for Python	classification_report

Model Data Flow Diagram

- Data Preprocessing: Utilizes TensorFlow's ImageDataGenerator for image resizing, normalization, and augmentation to enhance model generalizability and prevent overfitting.
- Train, Validation, and Test: The training data is used to train the model, validation data is used to tune the parameters
- Xception Model: A type of deep neural network model that is particularly good at tasks like image recognition.
- Training Results: After the model is built, it is trained using the training data, and the results or performance metrics of this training process are recorded.
- Testing Results: Finally, the trained Xception model is tested using the test data to evaluate its performance, and the results of this test are noted.



Model Evaluation Results

- High Precision and Recall: The model exhibits high precision and recall for clothes and furniture.
- Lower Performance for ‘Medical: Exhibits lower precision and recall in the ‘medical’ category
- Overall Robustness: Maintains an overall accuracy of 0.77 and a weighted average f1-score of 0.77, demonstrating strong and consistent performance across various classes.

Classification Report:

class	precision	recall	f1-score	support
biowaste	0.75	0.81	0.78	200
cardboard	0.74	0.72	0.73	150
clothes	0.87	0.89	0.88	220
e-waste	0.80	0.78	0.79	190
furniture	0.83	0.90	0.86	40
glass	0.76	0.76	0.76	150
medical	0.65	0.73	0.69	120
metal	0.68	0.78	0.73	170
paper	0.78	0.60	0.68	180
plastic	0.77	0.67	0.71	160
shoes	0.82	0.87	0.84	170
accuracy			0.77	1800
macro avg	0.77	0.77	0.77	1800
weighted avg	0.77	0.77	0.77	1800

InceptionResNetv2

Model Proposal

- Deep convolutional neural network (CNN) architecture that combines elements from the Inception and ResNet architectures with 164 layers.
- Introduced by Google researchers
- Residual Connections: The incorporation of residual connections allows for the training of very deep networks by addressing the vanishing gradient problem.
- Inception Modules: InceptionResNetV2 includes Inception modules, which consist of multiple convolutional pathways with different kernel sizes.

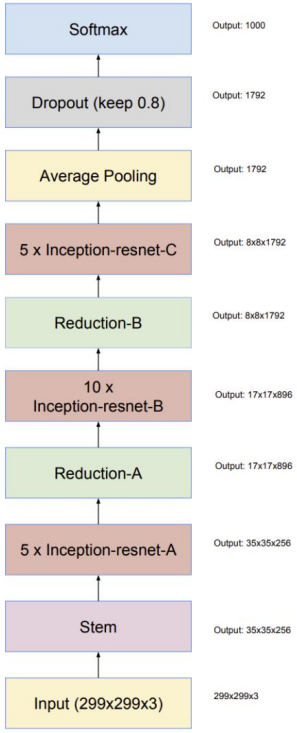


Figure 15. Schema for Inception-ResNet-v1 and Inception-ResNet-v2 networks. This schema applies to both networks but the underlying components differ. Inception-ResNet-v1 uses the blocks as described in Figures 14, 10, 7, 11, 12 and 13. Inception-ResNet-v2 uses the blocks as described in Figures 3, 16, 7,17, 18 and 19. The output sizes in the diagram refer to the activation vector tensor shapes of Inception-ResNet-v1.

Model Proposal

- The number of parameters in InceptionResNetV2 is around 55 million.
- The architecture consists of several units such as Inception modules, Residual Connections, Residual Blocks, Auxiliary Classifiers.
- Inception modules capture and process information from different spatial resolutions whereas reduction blocks use a combination of convolution and max-pooling to reduce spatial dimensions while increasing the number of filters which helps in plummeting computational complexity.

Reduction Module

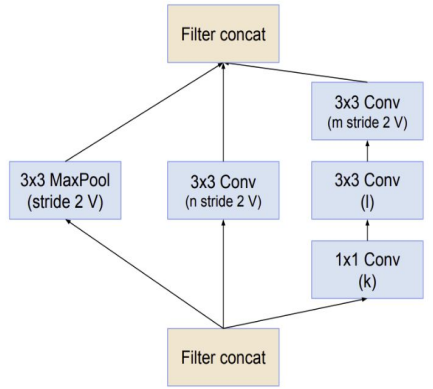


Figure 7. The schema for 35×35 to 17×17 reduction module. Different variants of this blocks (with various number of filters) are used in Figure 9, and 15 in each of the new Inception(-v4, -ResNet-v1, -ResNet-v2) variants presented in this paper. The k, l, m, n numbers represent filter bank sizes which can be looked up in Table 1.

Note. Source: (Szegedy et al, 2016)

Inception-ResNet-A module

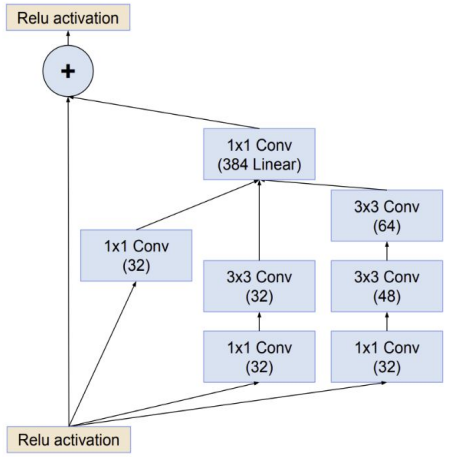


Figure 16. The schema for 35×35 grid (Inception-ResNet-A) module of the Inception-ResNet-v2 network.

Note. Source: (Szegedy et al, 2016)

Model Support

Hardware Configuration

Component	Specification
CPU	11th Generation Intel® CoreTM – i9-11950H Processor with vProTM (2.60 GHz, up to 5.00 GHz with Turbo Boost, 8 Cores, 16 Threads, 24 MB Cache)
GPU	NVIDIA® GeForce RTXTM 3080 with Max-Q 16GB
RAM	64 GB DDR4 3200MHz (2x32 GB)
Storage	1 TB Ple SSD Gen4

Software Configuration

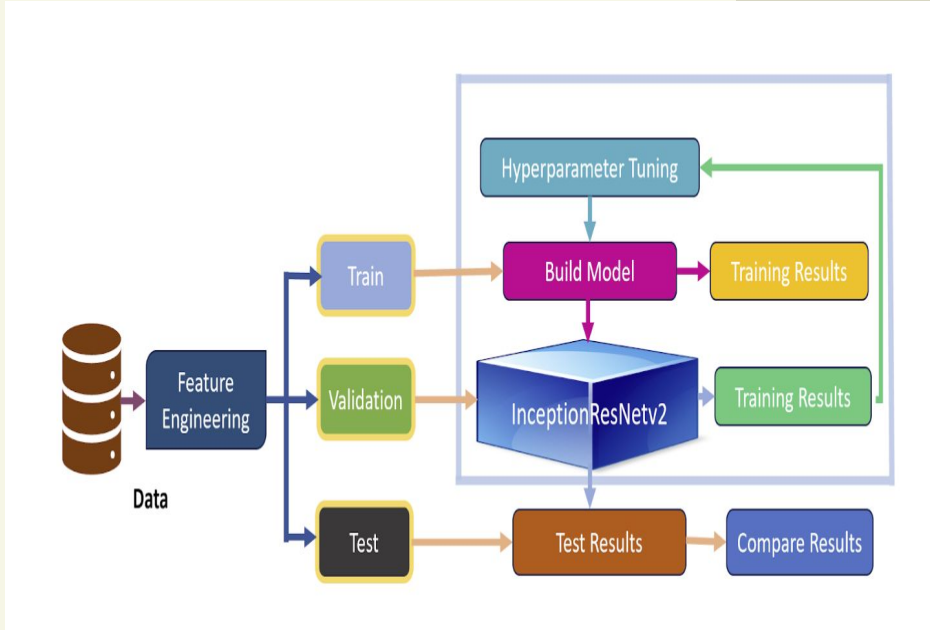
Component	Specification
OS	Windows 10 Pro 64
IDE	PyCharm 2022.2.2 (Edu)
Google Collaboratory Notebook Python	3.10.12

Tools

Library/Tool	Method	Usage
TensorFlow	Keras API, ImageDataGenerator, InceptionResNetV2, GlobalAveragePooling2D, Dense, Dropout, Model, EarlyStopping,ModelCheckpoint, ReduceLROnPlateau	For building InceptionResNetv2 architecture
Keras Tuner	RandomSearch	For Hyperparameter tuning
sklearn.metrics	classification_report, confusion_matrix	For plotting classification report and confusion matrix
Matplotlib.pyplot	plot	For plotting training and validation accuracies and losses
Numpy	Random_seed	For reproducibility

Model Data Flow Diagram

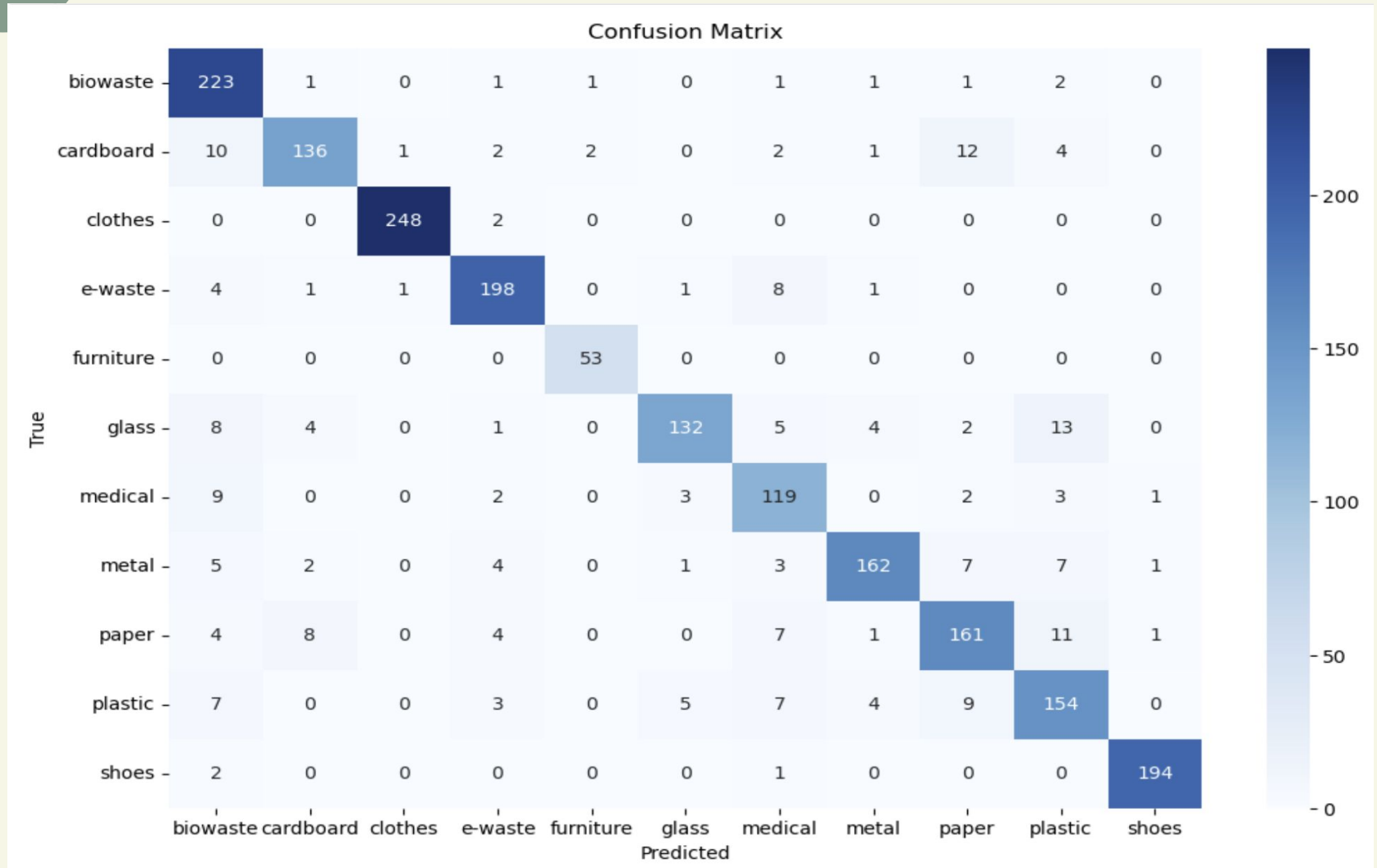
- ImageDataGenerator by the Keras library is used for real-time data augmentation during model training. It generates batches of augmented image data in real-time while the model is still being trained.
- Data generators feed batches of data to the model during training, allowing for a continuous flow of data without loading the entire dataset at once.
- RandomSearch from Keras tuner is used to perform hyperparameter tuning that samples sets from the hyperparametric search space.



Model Data Flow Diagram

- Global Average Pooling is applied to reduce the spatial dimensions of the feature maps and obtain a global representation. It helps in reducing the number of parameters in the model.
- Dropout Layer is used for regularization
- A dense hidden layer is added with varying hyperparametric searchable units.
- Output layer is an 11 class node with a softmax function.
- Callbacks such as EarlyStopping to stop training when a monitored metric has stopped improving, ModelCheckPoint to save the model's weights during training, and ReduceLROnPlateau to reduce the learning rate when a monitored metric has stopped improving is applied. This improves execution efficiency.
- Model is evaluated and accuracy and loss are extracted. Their respective plots are generated.
- Confusion Matrix and Classification reports are generated to analyze the results.

Model Evaluation Results



Model Evaluation Results

- The harmonic mean of Precision and Recall, F1-score performs the best for Furniture, Clothes, and Shoes.
- Macro average and Weighted average for all Precision, Recall, and F1-score are 88% showing consistent performance across all classes.
- The model can further be enhanced.

Class	Precision	Recall	F1-Score	Support
Biowaste	0.86	0.95	0.90	231
Cardboard	0.86	0.78	0.82	170
Clothes	0.98	0.97	0.98	250
E-waste	0.86	0.91	0.88	214
Furniture	0.96	1.00	0.98	53
Glass	0.90	0.82	0.85	169
Medical	0.73	0.86	0.79	139
Metal	0.93	0.79	0.85	192
Paper	0.82	0.76	0.79	197
Plastic	0.79	0.84	0.82	189
Shoes	0.97	0.98	0.98	197
Accuracy			0.88	2001
Macro Avg	0.88	0.88	0.88	2001
Weighted Avg	0.88	0.88	0.88	2001

YoloV8

Model Proposal

- YOLOv8 is a state of the art object detection model.
- It has recently included classification capabilities
- The model is pretrained on ImageNet dataset which is similar to all the other models of our research.
- As a one-stage detector, YOLO completes both object recognition and classification in a single network run
- Basic architecture is made up of Backbone, Neck, and Head Layers
- Backbone: Extracts high-level features from the input image.
- Neck: Refines and merges features obtained from the backbone.
- Produces bounding box predictions and class probabilities.

Model Proposal

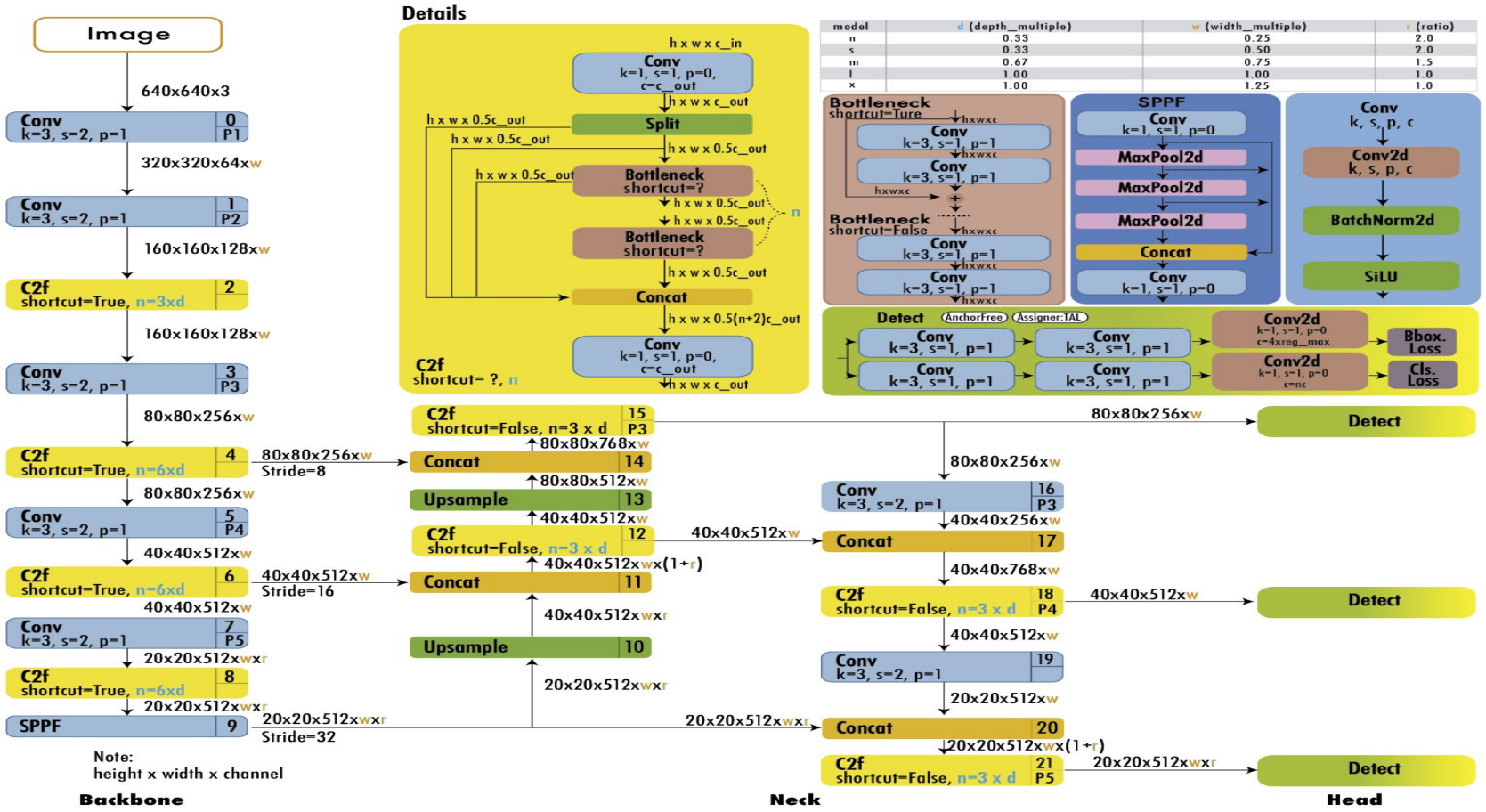


Figure 16: YOLOv8 Architecture. The architecture uses a modified CSPDarknet53 backbone. The C2f module replaces the CSPLayer used in YOLOv5. A spatial pyramid pooling fast (SPPF) layer accelerates computation by pooling features into a fixed-size map. Each convolution has batch normalization and SiLU activation. The head is decoupled to process objectness, classification, and regression tasks independently. Diagram based in [109].

Model Support

Hardware Configuration

Component	Specification
CPU	Intel® Xeon® CPU @ 2.00Ghz
GPU	A100 SXM4
RAM	40 GB
Storage	54 GB

Software Configuration

Component	Specification
OS	Windows 10 Pro 64
IDE	PyCharm 2022.2.2 (Edu)
Google Collaboratory Notebook Python	3.10.12

Tools

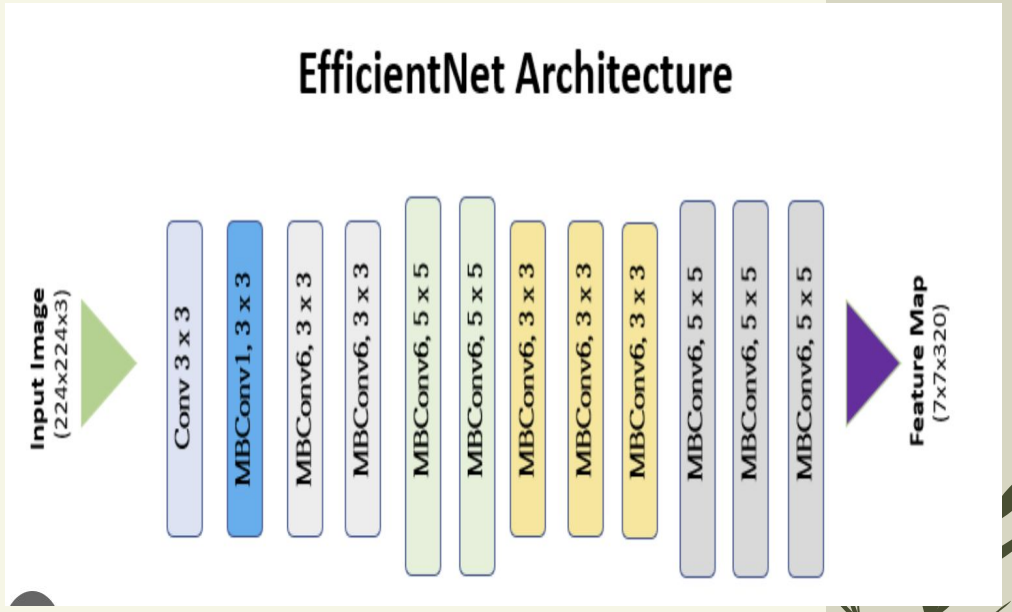
Library/Tool	Package	Method	Usage
Google colab	os	Getcwd, chdir, path.join	Get and change directories, navigating through different paths
	drive	mount	To mount drive
Ultralytics	YOLO	Train, val, and predict, task = classify, detect	Training ,validation, and predictions for classification and detection
	Glob	glob	For carrying list of all image paths
	IPython.display	Image	To view image from a path with specific size.
	Numpy	Random_seed	For reproducability
tensorboard		logdir	For monitoring training performance

Tools

Library/Tool	Method	Usage
TensorFlow	keras.applications.ResNet50, keras.layers.GlobalAveragePooling2D, keras.layers.Dense, keras.layers.Dropout, keras.Model, keras.callbacks.EarlyStopping, keras.callbacks.ModelCheckpoint, keras.callbacks.ReduceLROnPlateau	Primary deep learning framework
Keras Tuner	RandomSearch	For optimizing the hyperparameters of neural networks
Numpy	Random_seed	Array processing for numbers, strings, records, etc.

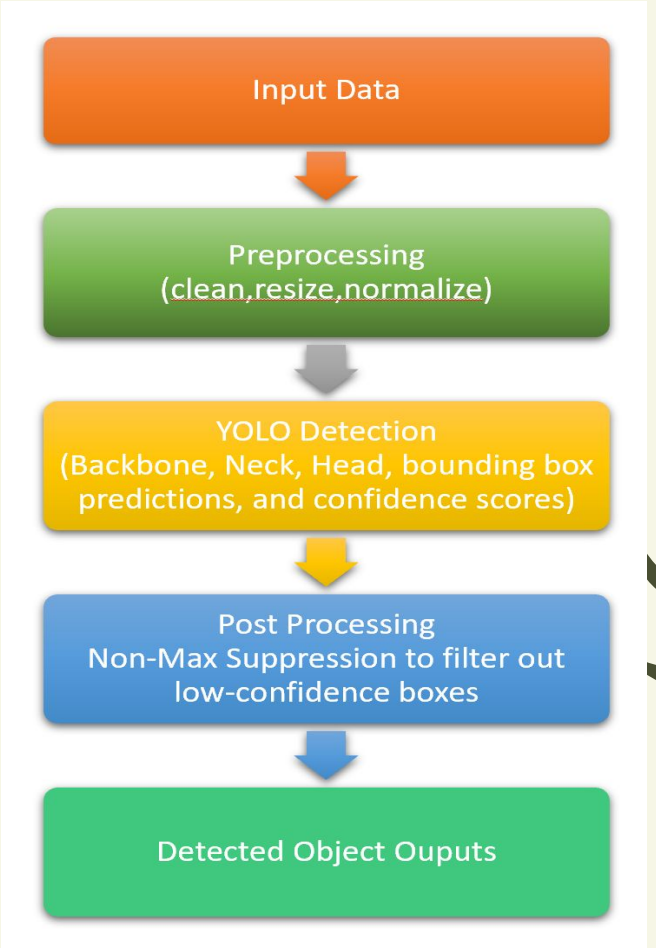
Model Data Flow Diagram

- It does classification using a version of the EfficientNet architecture for classification.
- The key idea is to balance the model size with accuracy.
- It comes in five flavors - nano, small, medium, large , extra large
- All the parameters can be configured using a yaml file.
- The model takes in task as classification or detection and mode to be train, val for validation and predict for inference.



Model Data Flow Diagram

- YOLO's Object Detection is trained on COCO dataset that has 90 categories.
- It is very efficient and predicts bounding box and confidence scores for target class.
- mAP, Precision, Recall, F1-curve are calculated to evaluate the performance.



Model Data - Classification

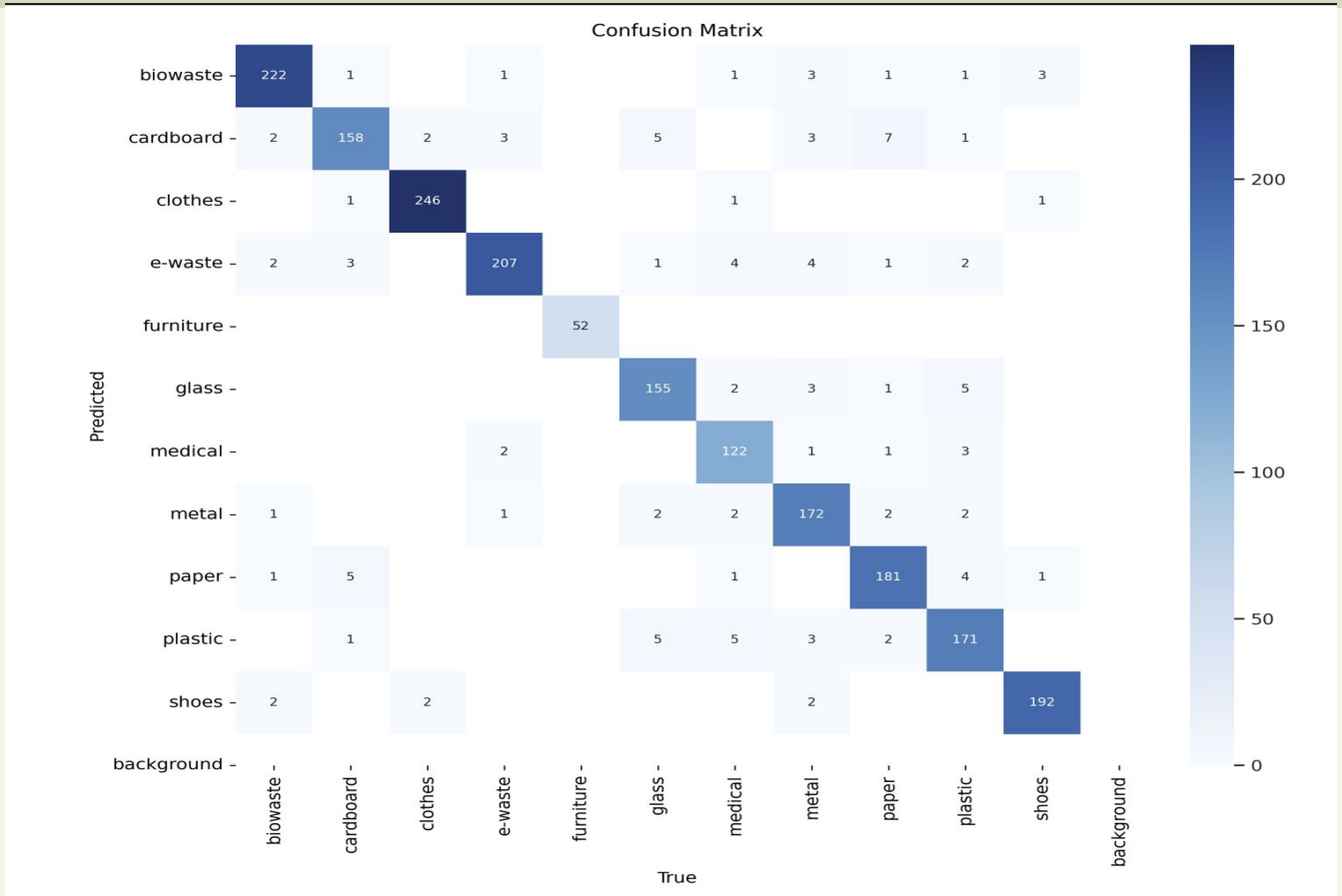
	precision	recall	f1-score	support
biowaste	0.97	0.94	0.95	231
cardboard	0.85	0.90	0.87	170
clothes	0.98	1.00	0.99	250
e-waste	0.92	0.89	0.91	214
furniture	1.00	1.00	1.00	53
glass	0.90	0.88	0.89	169
medical	0.88	0.88	0.88	139
metal	0.87	0.92	0.90	192
paper	0.86	0.89	0.87	197
plastic	0.93	0.87	0.90	189
shoes	0.97	0.96	0.97	197
accuracy			0.92	2001
macro avg	0.92	0.92	0.92	2001
weighted avg	0.92	0.92	0.92	2001

YOLOv8 - Nano

	precision	recall	f1-score	support
biowaste	0.97	0.96	0.96	231
cardboard	0.88	0.90	0.89	170
clothes	0.98	1.00	0.99	250
e-waste	0.93	0.93	0.93	214
furniture	0.96	1.00	0.98	53
glass	0.89	0.88	0.89	169
medical	0.95	0.87	0.91	139
metal	0.88	0.95	0.91	192
paper	0.88	0.90	0.89	197
plastic	0.91	0.86	0.89	189
shoes	0.98	0.98	0.98	197
accuracy			0.93	2001
macro avg	0.93	0.93	0.93	2001
weighted avg	0.93	0.93	0.93	2001

YOLOv8 - Small

YOLO Evaluation Results - Classification



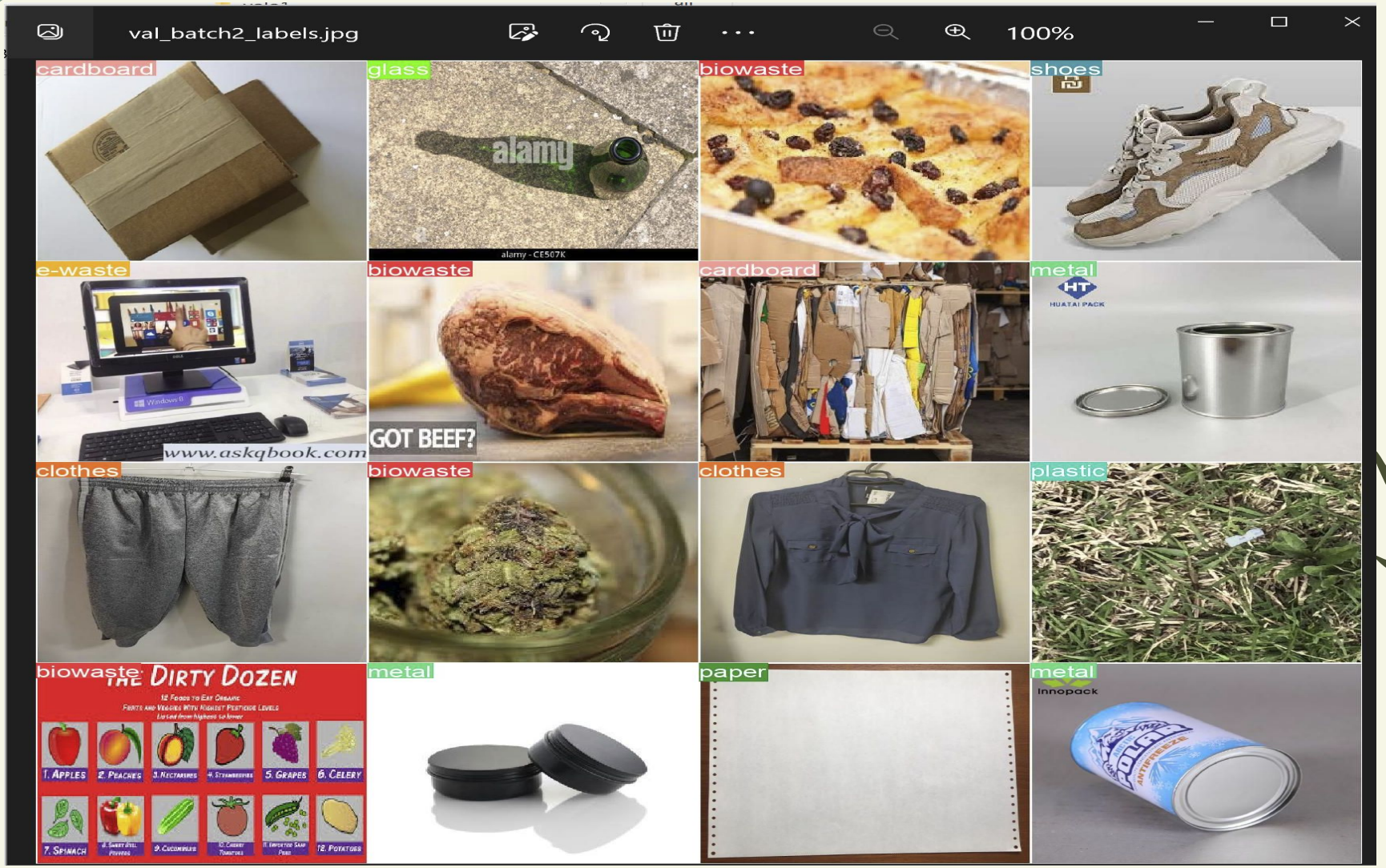
YOLOv8 - small

YOLO Evaluation Results - Classification

Model	Top1	Top5	Inference Speed
YOLOv8-n	93.1%	99.6%	0.6
YOLOv8-s	94%	99.6%	1.0

- YOLOv8-n nano model took 5.5 hours to train on 100 epochs
- YOLOv8-s small took 7.3 hours to train on 100 epochs
- YOLOv8-s small performed the best with highest accuracy of 94%

YOLO Evaluation Results – Classification

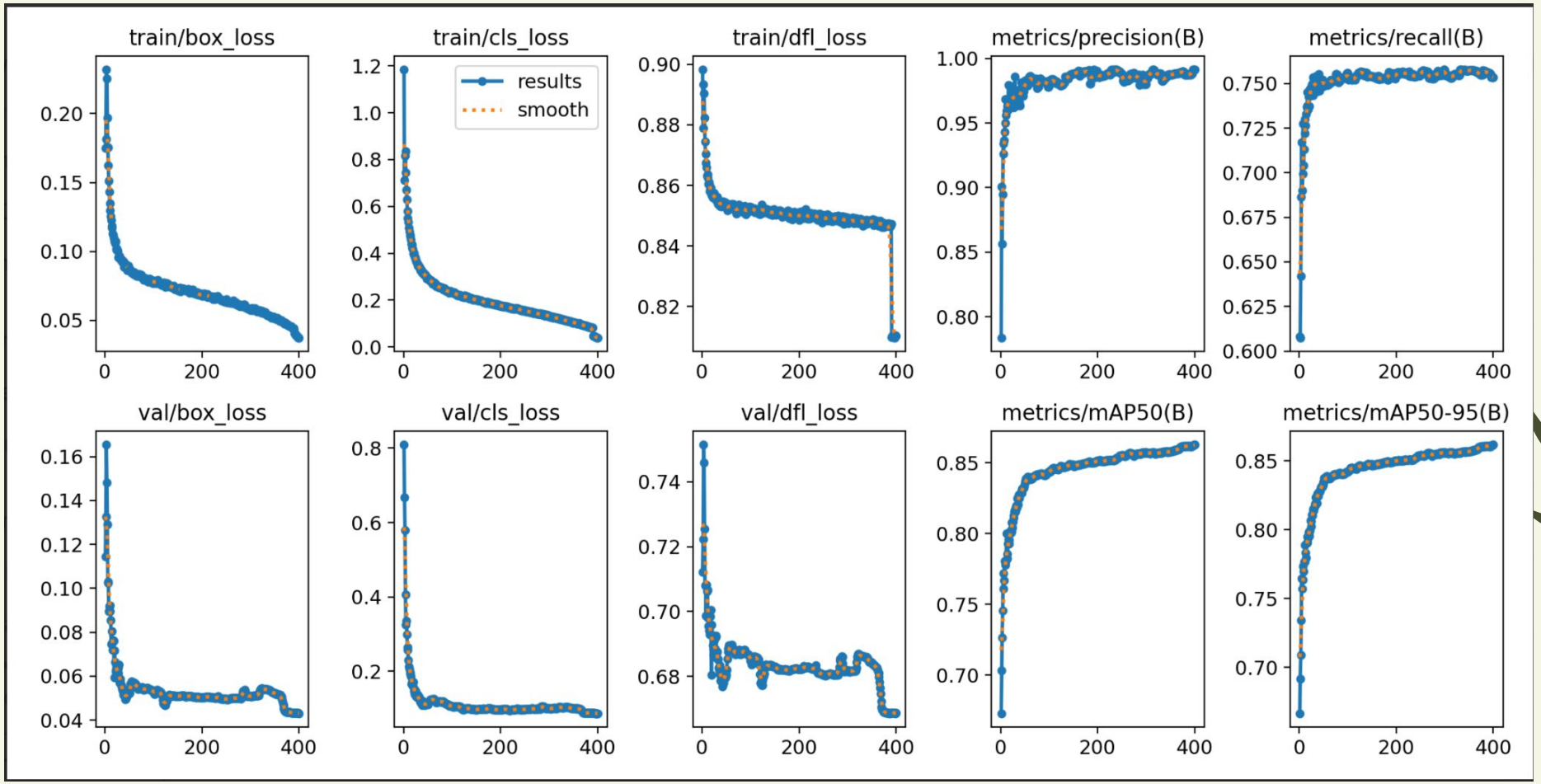


YOLOv8 Different Versions Model Comparison - Object Detection

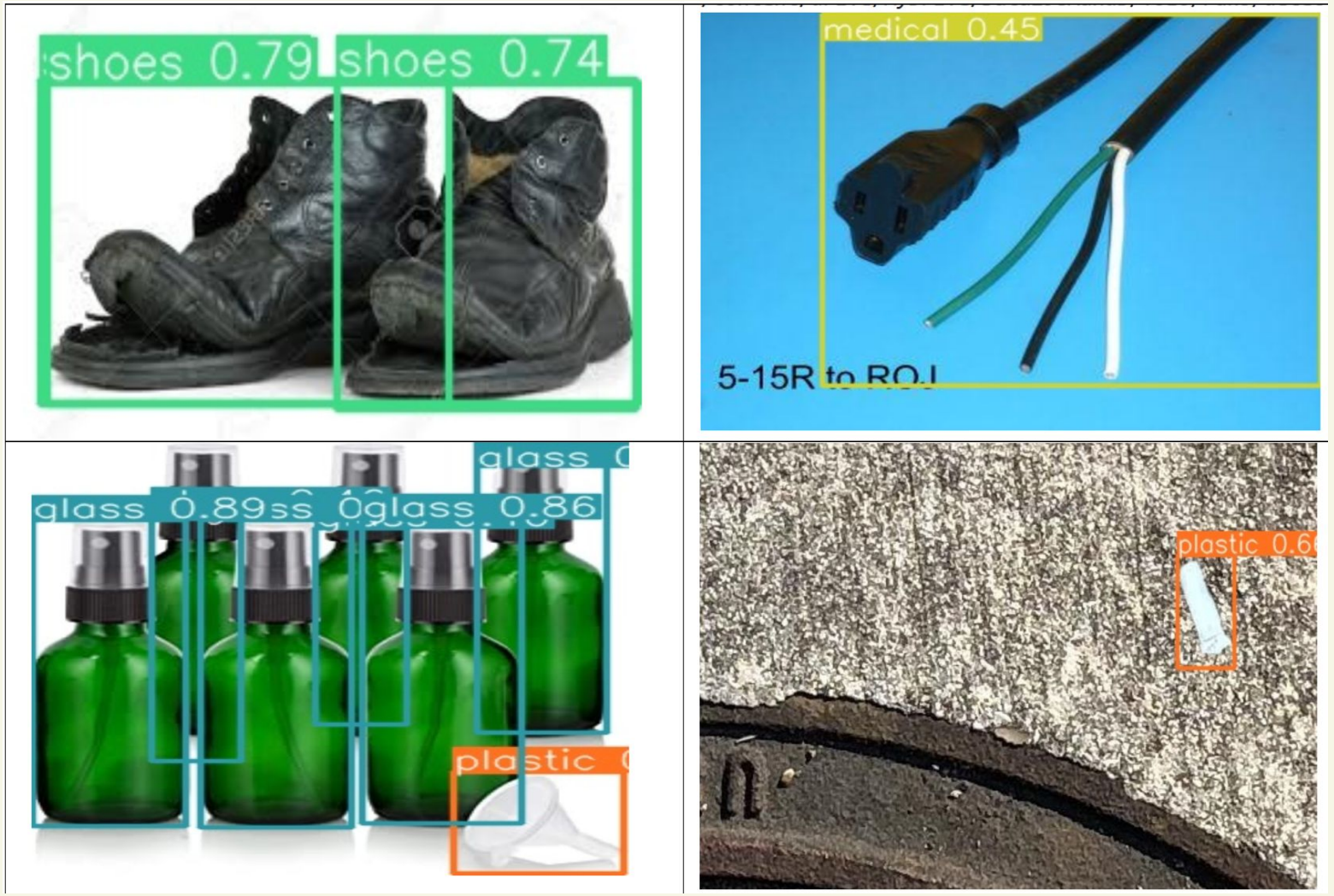
Model	mAP 50	mAP 50-95	Inference Time (ms)
YOLOv8-m (1100 images)	0.579	0.409	0.6
YOLOv8-m	0.745	0.699	1.7
YOLOv8-l	**0.863**	**0.862**	**4.6**

- Initially YOLOv8 medium(295 layers) model trained for 25 epochs on 1100 images annotated with CVAT tool
- Training took 0.055 hours
- Annotated images through Grounding DIno, more than 1100 images per class got detected.
- Trained on YOLOv8-medium again for 100 epochs.. e-waste category performed really badly.
- After working on the dataset again re-trained the model on YOLOv8's large variant for 400 epochs.
- The model performed consistently across all the thresholds.

YOLOv8 Object Detection



YOLOv8 Object Detection



Model Comparison – Classification

Model	F1-Score
ResNet50	0.91
Xception	0.87
MobileNetv2	0.87
InceptionResnetv2	0.88
YOLOv8-n	0.92
YOLO v8-s	0.93

Thank You!

Do you have any questions?