Warlens: Transfer Learning for Event Classification in Conflict Zones

by

Vaidik Manori(21BCE5043)

Kushagra Pareek (21BCE1506)

Ibrahim Shakir (21BRS1334)

Final Project Report Template

1. Introduction
   1.1. Project overviews
   1.2. Objectives
2. Project Initialization and Planning Phase
   2.1. Define Problem Statement
   2.2. Project Proposal (Proposed Solution)
   2.3. Initial Project Planning
3. Data Collection and Preprocessing Phase
   3.1. Data Collection Plan and Raw Data Sources Identified
   3.2. Data Quality Report
   3.3. Data Preprocessing
4. Model Development Phase
   4.1. Model Selection Report
   4.2. Initial Model Training Code, Model Validation and Evaluation Report
5. Model Optimization and Tuning Phase
   5.1. Tuning Documentation
   5.2. Final Model Selection Justification
6. Results
   6.1. Output Screenshots
7. Advantages & Disadvantages
8. Conclusion
9. Future Scope
10. Appendix
   10.1. Source Code
   10.2. GitHub & Project Demo Link

Project Initialization and Planning Phase

Date	11 July 2024
Team ID	SWTID1720012105
Project Name	WarLens: Transfer Learning for Event Classification in Conflict Zones
Maximum Marks	3 Marks

Define Problem Statements (Customer Problem Statement Template):

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love. A well-articulated customer problem statement allows you and your team to find the ideal solution for your customers' challenges. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

Reference: https://miro.com/templates/customer-problem-statement/

Example:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Humanitarian aid worker.	To quickly and accurately classify	Current methods are either too	Real-time and reliable data to make	Frustrated and concerned about the safety and efficiency of our operations

		events in conflict zones.	slow or not precise enough.	informed decisions and respond effectively.	

Initial Project Planning Template

Date	11 July 2024
Team ID	SWTID1720012105
Project Name	WarLens: Transfer Learning for Event Classification in Conflict Zones
Maximum Marks	4 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create a product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data Collection	USN-1	As a data scientist, I can access verified datasets from Amazon to ensure data diversity and reliability.	2	High	Vaidik Kushagra Ibrahim Sounil	3/7/2024	4/7/2024
Sprint-1	Data Preprocessing	USN-2	As a data scientist, I can preprocess the data to clean and structure it for model training.	1	High	Vaidik Kushagra	3/7/2024	4/7/2024

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
						Ibrahim Sounil		
Sprint-2	Model Development	USN-3	As a machine learning engineer, I can develop a collaborative filtering model to analyze user preferences for electronic products	2	High	Vaidik Kushagra Ibrahim Sounil	5/7/2024	8/7/2024
Sprint-2	Model Optimization	USN-4	As a machine learning engineer, I can optimize the recommendation model to enhance its accuracy and relevance.	2	Medium	Vaidik Kushagra	5/7/2024	8/7/2024
Sprint-3	Model Evaluation	USN-5	As a data scientist, I can evaluate the model's performance using appropriate metrics to ensure its effectiveness.	1	High	Vaidik Ibrahim	9/7/2024	10/7/2024
Sprint-3	Deployment	USN-6	As a software engineer, I can deploy the recommendation model into a production environment.	3	High	Vaidik	9/7/2024	10/7/2024

Project Initialization and Planning Phase

Date	11 July 2024
Team ID	SWTID1720012105
Project Title	WarLens: Transfer Learning for Event Classification in Conflict Zones
Maximum Marks	3 Marks

Project Proposal (Proposed Solution) template

This project proposal outlines a solution to address a specific problem. With a clear objective, defined scope, and a concise problem statement, the proposed solution details the approach, key features, and resource requirements, including hardware, software, and personnel.

Project Overview	
Objective	The primary objective of WarLens is to develop a machine learning model utilizing transfer learning techniques to accurately classify events in conflict zones. This will aid in providing timely and actionable intelligence to humanitarian organizations and policymakers.
Scope	• Development and training of a transfer learning model. • Integration of the model into a user-friendly interface for real-time event classification. • Validation and testing using historical and real-time data from conflict zones.
Problem Statement	
Description	Conflict zones often experience a wide range of events that require immediate attention and action. Current methods for event classification are often slow, inefficient, and lack the ability to adapt quickly to new data.
Impact	Enhance the speed and accuracy of event classification in conflict zones.
Proposed Solution	

Approach	• Data Collection: Gathering and preprocessing data from various sources, like kaggle,etc. • Model Development: Selecting and fine-tuning a pre-trained model for event classification. • Integration: Developing an interface for users to interact with the model and receive classifications in real-time. • Testing and Validation: Ensuring the model's accuracy and reliability through rigorous testing.
Key Features	• Transfer Learning: Utilizes pre-trained models to reduce training time and improve accuracy. • Real-time Classification: Provides instant event classification, critical for timely decision-making. • User-Friendly Interface: Ensures ease of use for non-technical users such as humanitarian workers. • Scalability: Designed to handle increasing amounts of data and adapt to new types of events. • Comprehensive Data Sources: Incorporates diverse data inputs for a holistic view of events.

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	2 x NVIDIA V100 GPUs
Memory	RAM specifications	16 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	Tensorflow
Development Environment	IDE, version control	Google Collab notebook, Git
Data		

Data	Source, size, format	Kaggle dataset, 84,151,603 images

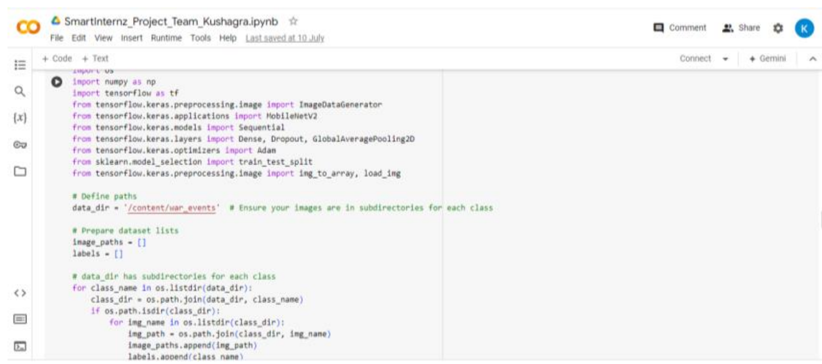
Data Collection and Preprocessing Phase

Date	11 July 2024
Team ID	SWTID1720012105
Project Title	WarLens: Transfer Learning for Event Classification in Conflict Zones
Maximum Marks	6 Marks

Preprocessing Template

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	**Description**
Data Overview	We are using a Kaggle dataset name war events with over 84,151,000 images which are having various types like fire,combat, DestroyedBuildings ,humanatarian ai and military vehicles and weapons.
Resizing	Resize images to a specified target size.
Normalization	Normalize pixel values to a specific range.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.
Denoising	Apply denoising filters to reduce noise in the images.
Edge Detection	Apply edge detection algorithms to highlight prominent edges in the images.

<table>
<tr><td>Color Space Conversion</td><td>Convert images from one color space to another.</td></tr>
<tr><td>Image Cropping</td><td>Crop images to focus on the regions containing objects of interest.</td></tr>
<tr><td>Batch Normalization</td><td>Apply batch normalization to the input of each layer in the neural network.</td></tr>
<tr><td colspan="2">Data Preprocessing Code Screenshots</td></tr>
<tr><td>Loading Data</td><td>
</td></tr>
<tr><td>Resizing</td><td>
</td></tr>
<tr><td>Normalization</td><td>
</td></tr>
</table>

<table>
<tr>
<td>Data Augmentation</td>
<td>
</td>
</tr>
<tr>
<td>Denoising</td>
<td>
</td>
</tr>
<tr>
<td>Edge Detection</td>
<td>
</td>
</tr>
<tr>
<td>Color Space Conversion</td>
<td>
</td>
</tr>
</table>

<table>
<tr><td>Image Cropping</td><td>
•</td></tr>
<tr><td>Batch Normalization</td><td></td></tr>
</table>

Data Collection and Preprocessing Phase

Date	11 July 2024
Team ID	SWTID1720012105
Project Title	WarLens: Transfer Learning for Event Classification in Conflict Zones
Maximum Marks	2 Marks

Data Quality Report Template

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

Data Source	Data Quality Issue	Severity	Resolution Plan
Dataset: Kaggle Dataset with 84,151,603 images	Incorrect Labels	High	Perform manual and automated validation of labels. Use a subset of images for manual verification and employ a model trained on a smaller verified dataset to predict and cross-check labels. Mislabeled images will be corrected or removed.
......(same dataset as above)	Imbalanced Classes	Moderate	Use techniques like data augmentation to balance the class distribution. This can involve

			generating new images for underrepresented classes through transformations such as rotation, flipping, and cropping.

Data Collection and Preprocessing Phase

Date	18 July 2024
Team ID	SWTID1720012105
Project Title	WarLens: Transfer Learning for Event Classification in Conflict Zones
Maximum Marks	2 Marks

Data Collection Plan & Raw Data Sources Identification

WarLens aims to gather and curate multimedia data (images and videos) from conflict zones to train and validate transfer learning models for event classification. The project focuses on high-risk areas identified through conflict maps, news reports, and satellite data. Key data sources include open-source intelligence (OSINT), social media platforms (Twitter, YouTube, Instagram), news agencies, NGOs, and satellite imagery providers. The data types include images and videos collected via web scraping tools from social media, news sites, and satellite imagery providers.

For WarLens, raw data will be sourced from diverse platforms. Images and videos will be gathered from social media platforms (Twitter, YouTube, Instagram), reputable news agencies, and satellite imagery providers. Additional sources include OSINT reports and NGO databases that provide real-time updates on conflict zones. The curated data will be systematically stored, ensuring data integrity and facilitating effective training of transfer learning models for accurate event classification in conflict zones.

Data Collection Plan

Section	Description
Project Overview	WarLens is an innovative machine learning project that utilizes transfer learning techniques to classify events in conflict zones by analyzing multimedia data such as images and videos. The project's objective is to enhance situational awareness and provide accurate event classification in high-risk areas.
Data Collection Plan	For WarLens, data will be collected from various sources, including social media platforms (Twitter, YouTube, Instagram), reputable news agencies, satellite imagery providers, and open source intelligence (OSINT) reports. This diverse data set will provide a comprehensive foundation for training transfer learning models to accurately classify events in conflict zones.
Raw Data Sources Identified	For WarLens, raw data will be sourced from social media platforms like Twitter, YouTube, and Instagram, which provide real-time user-generated content. Additional sources include reputable news agencies for verified multimedia reports, satellite imagery providers for detailed overhead views, OSINT reports for comprehensive conflict data, and NGOs for reliable on-the-ground information.

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Dataset	The data for WarLens consists of images and videos from social media platforms, verified multimedia reports from news agencies, detailed satellite imagery, comprehensive OSINT reports, and reliable on-the-ground content from NGOs.	https://drive.google.com/file/d/1_qiE733RgeD5f81AMal6scE_u2s4Tjza/view?usp=drivesdk	Image	84 MB	Private (with access)

Model Development Phase Template

Date	18 July 2024
Team ID	SWTID1720012105
Project Title	WarLens: Transfer Learning for Event Classification in Conflict Zones
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

Paste the screenshot of the model training code

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics

<table>
<tr>
<td>Model 1:
ResNet 50</td>
<td>
<pre><code># Load pre-trained ResNet50 model + higher level layers
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze convolutional layers
for layer in base_model.layers:
    layer.trainable = False

# Create a new model on top
model = Sequential([
    base_model,
    Flatten(),
    Dense(256, activation='relu'),
    Dense(len(label_to_index), activation='softmax') # Adjust the number of classes dynamically
])</code></pre>
</td>
<td>
<pre><code>Epoch 1/20
12/12 [==============================] - 139s 11s/step - loss: 15.9469 - accuracy: 0.2005 - val_loss: 4.5750 - val_accuracy: 0.1979
Epoch 2/20
12/12 [==============================] - 110s 9s/step - loss: 2.6478 - accuracy: 0.3152 - val_loss: 1.6185 - val_accuracy: 0.4559
Epoch 3/20
12/12 [==============================] - 108s 9s/step - loss: 1.7286 - accuracy: 0.3723 - val_loss: 1.4072 - val_accuracy: 0.3824
Epoch 4/20
12/12 [==============================] - 107s 9s/step - loss: 1.4118 - accuracy: 0.3777 - val_loss: 1.3793 - val_accuracy: 0.4265
Epoch 5/20
12/12 [==============================] - 103s 9s/step - loss: 1.3041 - accuracy: 0.4674 - val_loss: 1.2268 - val_accuracy: 0.4559
Epoch 6/20
12/12 [==============================] - 110s 9s/step - loss: 1.3147 - accuracy: 0.4647 - val_loss: 1.4095 - val_accuracy: 0.3824
Epoch 7/20
12/12 [==============================] - 106s 9s/step - loss: 1.3526 - accuracy: 0.4348 - val_loss: 1.4803 - val_accuracy: 0.3529
Epoch 8/20
12/12 [==============================] - 105s 9s/step - loss: 1.3010 - accuracy: 0.4783 - val_loss: 1.2002 - val_accuracy: 0.4559
Epoch 9/20
12/12 [==============================] - 101s 8s/step - loss: 1.3379 - accuracy: 0.4402 - val_loss: 1.1594 - val_accuracy: 0.5882
Epoch 10/20
12/12 [==============================] - ETA: 0s - loss: 1.2437 - accuracy: 0.4810Epoch 11/20
12/12 [==============================] - 99s 8s/step - loss: 1.2875 - accuracy: 0.5054 - val_loss: 1.1221 - val_accuracy: 0.5147
Epoch 12/20
12/12 [==============================] - 98s 8s/step - loss: 1.2368 - accuracy: 0.4538 - val_loss: 1.1342 - val_accuracy: 0.5588
Epoch 13/20
12/12 [==============================] - 102s 9s/step - loss: 1.2059 - accuracy: 0.5109 - val_loss: 1.1484 - val_accuracy: 0.5588
Epoch 14/20
12/12 [==============================] - 107s 9s/step - loss: 1.1280 - accuracy: 0.5547 - val_loss: 1.2937 - val_accuracy: 0.4853
Epoch 15/20
12/12 [==============================] - 103s 9s/step - loss: 1.2020 - accuracy: 0.5217 - val_loss: 1.0901 - val_accuracy: 0.5882
Epoch 16/20
12/12 [==============================] - 106s 9s/step - loss: 1.1269 - accuracy: 0.5625 - val_loss: 1.0758 - val_accuracy: 0.6029
Epoch 17/20
12/12 [==============================] - 104s 9s/step - loss: 1.1766 - accuracy: 0.5217 - val_loss: 1.1638 - val_accuracy: 0.5294
Epoch 18/20
12/12 [==============================] - 100s 8s/step - loss: 1.2492 - accuracy: 0.5272 - val_loss: 1.0006 - val_accuracy: 0.5735
Epoch 19/20
12/12 [==============================] - 100s 8s/step - loss: 1.1108 - accuracy: 0.5625 - val_loss: 1.2997 - val_accuracy: 0.4706
Epoch 20/20
12/12 [==============================] - 99s 8s/step - loss: 1.0935 - accuracy: 0.5897 - val_loss: 0.9670 - val_accuracy: 0.6324</code></pre>
</td>
</tr>
<tr>
<td>Model 2:

MobileNet</td>
<td>
<pre><code># Load pre-trained MobileNetV2 model + higher level layers
mobilenet_model = MobileNetV2(input_shape=(224, 224, 3), include_top=False, weights='imagenet')

# Freeze the pretrained layers
mobilenet_model.trainable = False

# Create a new model on top
model_new1 = Sequential([
    mobilenet_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(label_to_index), activation='softmax') # Adjust the number of classes dynamically
])

Compile the model
model_new1.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])</code></pre>
</td>
<td>
<pre><code>Epoch 1/20
12/12 [==============================] - 40s 3s/step - loss: 1.4525 - accuracy: 0.4844 - val_loss: 0.5592 - val_accuracy: 0.8229
Epoch 2/20
12/12 [==============================] - 30s 3s/step - loss: 0.7163 - accuracy: 0.7663 - val_loss: 0.4132 - val_accuracy: 0.8676
Epoch 3/20
12/12 [==============================] - 28s 2s/step - loss: 0.4721 - accuracy: 0.8397 - val_loss: 0.2712 - val_accuracy: 0.8971
Epoch 4/20
12/12 [==============================] - 28s 2s/step - loss: 0.3550 - accuracy: 0.8750 - val_loss: 0.2519 - val_accuracy: 0.8824
Epoch 5/20
12/12 [==============================] - 29s 2s/step - loss: 0.2780 - accuracy: 0.9212 - val_loss: 0.2339 - val_accuracy: 0.8824
Epoch 6/20
12/12 [==============================] - 29s 2s/step - loss: 0.3039 - accuracy: 0.9022 - val_loss: 0.2461 - val_accuracy: 0.9118
Epoch 7/20
12/12 [==============================] - 29s 3s/step - loss: 0.2404 - accuracy: 0.9212 - val_loss: 0.2074 - val_accuracy: 0.9118
Epoch 8/20
12/12 [==============================] - 27s 2s/step - loss: 0.2364 - accuracy: 0.9158 - val_loss: 0.2653 - val_accuracy: 0.8824
Epoch 9/20
12/12 [==============================] - 26s 2s/step - loss: 0.2033 - accuracy: 0.9375 - val_loss: 0.2496 - val_accuracy: 0.9118
Epoch 10/20
12/12 [==============================] - 27s 2s/step - loss: 0.1587 - accuracy: 0.9457 - val_loss: 0.2386 - val_accuracy: 0.8971
Epoch 11/20
12/12 [==============================] - 31s 3s/step - loss: 0.2181 - accuracy: 0.9158 - val_loss: 0.1994 - val_accuracy: 0.8971
Epoch 12/20
12/12 [==============================] - 30s 2s/step - loss: 0.1696 - accuracy: 0.9457 - val_loss: 0.2491 - val_accuracy: 0.8971
Epoch 13/20
12/12 [==============================] - 28s 2s/step - loss: 0.1672 - accuracy: 0.9429 - val_loss: 0.2166 - val_accuracy: 0.9118
Epoch 14/20
12/12 [==============================] - 27s 2s/step - loss: 0.1654 - accuracy: 0.9401 - val_loss: 0.2081 - val_accuracy: 0.8971
Epoch 15/20
12/12 [==============================] - 29s 2s/step - loss: 0.1147 - accuracy: 0.9647 - val_loss: 0.2717 - val_accuracy: 0.8824
Epoch 16/20
12/12 [==============================] - 29s 2s/step - loss: 0.1250 - accuracy: 0.9511 - val_loss: 0.2085 - val_accuracy: 0.9118
Epoch 17/20
12/12 [==============================] - 38s 3s/step - loss: 0.1561 - accuracy: 0.9592 - val_loss: 0.2044 - val_accuracy: 0.8824
Epoch 18/20
12/12 [==============================] - 33s 3s/step - loss: 0.1446 - accuracy: 0.9511 - val_loss: 0.2180 - val_accuracy: 0.9265
Epoch 19/20
12/12 [==============================] - 29s 3s/step - loss: 0.1043 - accuracy: 0.9647 - val_loss: 0.2229 - val_accuracy: 0.8971
Epoch 20/20
12/12 [==============================] - 25s 2s/step - loss: 0.1076 - accuracy: 0.9647 - val_loss: 0.2015 - val_accuracy: 0.9265</code></pre>
</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</table>

Model Development Phase Template

Date	18 July 2024
Team ID	SWTID1720012105
Project Title	WarLens: Transfer Learning for Event Classification in Conflict Zones
Maximum Marks	5 Marks

Model Selection Report

In the model selection report for future deep learning and computer vision projects, various architectures, such as CNNs or RNNs, will be evaluated. Factors such as performance, complexity, and computational requirements will be considered to determine the most suitable model for the task at hand.

Model Selection Report:

Model	**Description**
Resnet50	It is a deep convolutional neural network designed to address the vanishing gradient problem in training deep networks. It achieves this by introducing residual connections, allowing gradients to flow directly through layers, bypassing intermediate layers. ResNet50 is widely used for image classification tasks due to its ability to learn deep representations and its robust performance across various datasets.
MobileNetV2	MobileNetV2 is a convolutional neural network architecture optimized for mobile and embedded vision applications. It uses depthwise separable convolutions to significantly reduce the number of parameters and computational complexity. MobileNetV2 also introduces inverted residuals

	and linear bottlenecks, enhancing the network's efficiency and accuracy. This model is particularly suited for resource-constrained environments while maintaining high performance in image classification tasks.

Model Optimization and Tuning Phase Template

Date	19 July 2024
Team ID	SWTID1720012105
Project Title	WarLens: Transfer Learning for Event Classification in Conflict zones.
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters

Model 1	`# Train the model`
	`model.fit(`
	`    train_gen,`
	`    steps_per_epoch=len(train_paths) // batch_size,`
	`    validation_data=val_gen,`
	`    validation_steps=len(val_paths) // batch_size,`
	`    epochs=20`
	`)`
	
	`# Save the model`
	`model.save('war_lens_model_resnet50.h5')`

<table>
<tr>
<td></td>
<td>

```
# Create a new model on top
model = Sequential([
    base_model,
    Flatten(),
```

```
    Dense(256, activation='relu'),
    Dense(len(label_to_index), activation='softmax')  # Adjust the number of classes dynamically
])
```

</td>
</tr>
<tr>
<td>Model 2</td>
<td>

```
# Train the model
history_new1 = model_new1.fit(
    train_gen,
    steps_per_epoch=len(train_paths) // batch_size,
    validation_data=val_gen,
    validation_steps=len(val_paths) // batch_size,
    epochs=20
)

# Save the model
model_new1.save('war_lens_model_mobilenetv2.h5')
```

```
# Create a new model on top
model_new1 = Sequential([
    mobilenet_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(label_to_index), activation='softmax')  # Adjust the number of classes dynamically
])

# Compile the model
```

</td>
</tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning

Model 2	• **Model Efficiency:** ◦ **ResNet50:** The ResNet50 model includes a Flatten layer, which results in a large number of parameters, potentially leading to overfitting and higher computational cost. ◦ **MobileNetV2:** The MobileNetV2 model uses GlobalAveragePooling2D, which reduces the number of parameters and makes the model more efficient and less prone to overfitting. • **Regularization:** ◦ **ResNet50:** No explicit regularization layer is added. ◦ **MobileNetV2:** Includes a Dropout layer with a 50% drop rate, which helps prevent overfitting by randomly setting half of the units to zero during training. • **Model Complexity:** ◦ **ResNet50:** The model is deeper and more complex, which can make it more challenging to train and tune properly. ◦ **MobileNetV2:** The model is designed to be lightweight and efficient, making it easier to train and less likely to overfit, especially with limited data. • **Parameter Tuning:** ◦ **ResNet50:** The dense layer with 256 units might not be optimal for your dataset, potentially leading to overfitting or underfitting. ◦ **MobileNetV2:** The dense layer with 128 units, combined with dropout, strikes a balance between model complexity and generalization ability. **Epochs:** ◦ Both models are trained for 20 epochs, which should be sufficient for convergence. However, MobileNetV2’s efficient architecture might allow it to converge to a better minimum within the same number of epochs.

Conclusion

In this project, we successfully employed various deep learning architectures, including ResNet50, MobileNet, Inception, and Xception, to accurately classify different images. The comparative analysis of these models has demonstrated their effectiveness and robustness in image classification tasks. Each architecture's unique design and capabilities contributed to achieving high accuracy in our classifications.

The results underscore the potential of deep learning techniques in tackling complex image classification problems and pave the way for further exploration and optimization of these models for enhanced performance.

by

Vaidik Manori(21BCE5043)

Kushagra Pareek (21BCE1506)

Ibrahim Shakir (21BRS1334)