Image Classification Model

Alap Dhruva (400590512)
Computing and Software
McMaster
Hamilton, Canada

***Abstract*— This report presents the implementation and evaluation of a fusion model approach to image classification, addressing the challenges of combining multiple models trained on different or overlapping class subsets. In Task A, three independent classifiers were trained on distinct class groups, and their outputs were integrated using feature-level fusion to improve overall classification accuracy. In Task B, the models were trained on overlapping class subsets to simulate more complex multi-domain learning scenarios. The effectiveness of the fusion model was analyzed, showing an improvement in generalization and classification performance.**

I. Introduction

Image classification, a core computer vision task, has seen tremendous progress with deep learning, especially Convolutional Neural Networks (CNNs). CNNs automatically learn hierarchical features from images, outperforming traditional methods. However, single CNN models face difficulties when dealing with diverse datasets, encountering issues like catastrophic forgetting and poor generalization.

This project addresses these challenges by employing a feature-level fusion strategy. Instead of relying on a single all-encompassing model, we combine multiple specialized CNNs, each trained on specific subsets of the data by fusing the feature representations learned by these individual models, the system aims to achieve improved classification accuracy and robustness, particularly in complex, multi-domain scenarios.

II. Design

A. TASK A

The Task A is consisting of 3 models followed by fusion model, each model focus on training 5 distinct classes. Task A dataset derived from CIFAR - 100 dataset. All three of the models follows similar architecture design but having a difference in filter size, drop out rate. The building blocks of the three models are shown in the diagram.

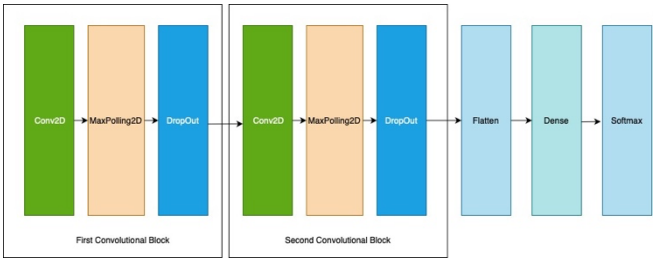

Fig. 1. Architecture of the TASK A models

All task A models process images through a series of layers that learn hierarchical features. First, convolutional layers use filters to detect spatial patterns, activation functions like ReLU introduce non-linearity, pooling layers down sample feature maps for robustness and efficiency, dropout layers prevent overfitting by randomly deactivating neurons, the flatten layer converts feature maps into a vector, dense layers learn high-level relationships for classification, and the final softmax layer outputs a probability distribution over the predicted classes.

The first model architecture is a baseline model designed with simplicity and efficiency in mind, incorporating two convolutional layers for feature extraction. The first layer uses 32 filters with a 3×3 kernel and ReLU activation, followed by max pooling to down sample the spatial dimensions, and dropout with a rate of 0.25 for regularization. The second convolutional layer has 128 filters, maintaining the same kernel size and activation function, paired with another max pooling and dropout layer to enhance generalization. The architecture flattens the data into a dense layer with 128 neurons to aggregate features before passing the output to a softmax layer for class probabilities. With its balanced design of depth and regularization, this model effectively prevents overfitting, making it ideal for small to medium datasets while ensuring computational efficiency.

The second model architecture enhances the baseline by increasing the filters in the first convolutional layer to 48, improving the model's ability to capture finer details in the input. This model uses a higher dropout rate of 0.3 to strengthen regularization and reduce the risk of overfitting. The second convolutional layer retains 128 filters for deeper feature extraction, followed by max pooling and dropout to balance performance and generalization. Like the baseline, the architecture flattens the data before passing it through a dense layer with 128 neurons and a final softmax layer for classification. These adjustments allow the model to handle more complex noises, offering improved feature representation without significantly increasing computational demands.

The third model architecture prioritizes generalization by introducing a higher number of filters (64) in the first convolutional layer and implementing early dropout (0.3) immediately after. This approach ensures that the model begins regularization early, reducing reliance on specific neurons and improving robustness. The second convolutional layer uses 128 filters to balance depth and efficiency, followed by max pooling and another dropout layer to manage overfitting. The flattened features are processed through a dense layer with 128 neurons for robust feature aggregation, with the final softmax layer outputting class probabilities. This model is tailored for datasets with high variability or limited samples, achieving a strong balance between feature extraction, regularization, and computational efficiency.

The use of two convolutional blocks in this architecture achieves an effective balance between feature extraction, computational efficiency, and regularization, making it a strong choice for achieving good accuracy. The first block captures low-level features like edges, while the second builds on these to identify more complex patterns, enabling robust hierarchical learning. By including max pooling for

dimensionality reduction and dropout for regularization, the design prevents overfitting while ensuring translation invariance and generalization. The fully connected layer aggregates features for robust classification, and the final softmax activation provides interpretable outputs for multi-class tasks. This architecture, with its simplicity and focus on essential patterns, ensures accurate and efficient performance across diverse datasets.

B. Task B

The task B dataset is derived from ILSVRC 2012 and also for model 1 and 2, I used additional dataset derived from ImageNet. Task B also consist of 3 models followed by fusion model. The merge model is the vector of 12 classes.

I have used couple of enhanced feature extraction block to increase the accuracy of the model 2 and 3. The purpose of enhanced extraction block is to extract meaningful features through separable convolutions, normalizing activations, and improving generalization using dropout, while also reducing spatial dimensions with max pooling for computational efficiency. Please refer the figure 1.2 to see single enhanced feature extraction block.

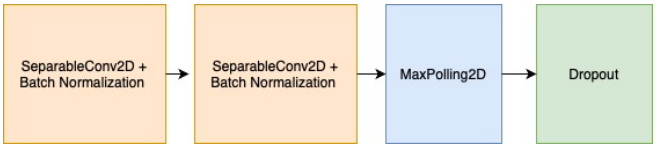

Fig. 2. Single block of feature extraction

This block performs efficient feature extraction by applying separable convolutions to capture spatial features with fewer parameters, followed by batch normalization to stabilize and accelerate training. Max pooling reduces spatial dimensions, focusing on dominant features, while dropout adds regularization to prevent overfitting.

The model 3 of task B architecture is the same as shown the figure 1. Please refer the below figure to see the design of models 1 and 2 architecture.

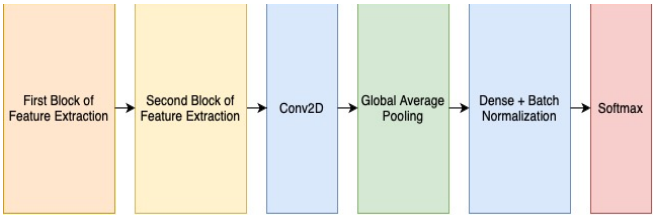

Fig. 3. Model 1 and 2 Architecture design – Task B

The model 1 architecture leverages separable convolutions for computational efficiency and deeper feature extraction. In Block 1, the model uses wider filters (96) to capture diverse features such as edges and patterns specific to the input dataset, followed by batch normalization for stability and dropout for regularization. Block 2 increases the filter count to 192, focusing on extracting more complex textures and patterns while maintaining regularization with higher dropout. A convolutional layer with 384 filters further aggregates features, reducing spatial dimensions with global average pooling to generate a compact representation. The classifier consists of a dense layer for feature refinement, followed by a batch normalization layer, and concludes with a softmax output layer for multi-class classification. This design ensures robust learning with balanced generalization and precision, particularly suited for datasets with intricate patterns.

The model 2 architecture also emphasizes efficiency through separable convolutions with narrower filters. Block 1 employs 64 filters to extract foundational features with reduced computational cost, complemented by batch normalization for convergence and dropout for overfitting prevention. Block 2 doubles the filter count to 128, utilizing depthwise separable convolutions to capture intermediate-level patterns, supported by max pooling for dimensionality reduction and a moderate dropout rate. A final feature extraction block uses a standard convolutional layer with 256 filters, followed by global average pooling and a higher dropout rate to ensure robust generalization. The dense layer with 256 units refines the features, and batch normalization ensures stability before the softmax output layer. This architecture prioritizes efficiency and regularization.

The 3rd model is designed with simplicity and efficiency, focusing on achieving generalization through strategic dropout and controlled complexity. The initial convolutional block utilizes 64 filters, coupled with max pooling for spatial reduction and early dropout at 30%, to mitigate overfitting right from the start. The second block doubles the filter count to 128 for capturing more complex features, maintaining max pooling and dropout for consistency in regularization. After feature extraction, the architecture flattens the output to a dense layer with 256 units, enabling effective representation learning. The final softmax layer provides multi-class predictions.

C. Fusion/Merged Model – Task A

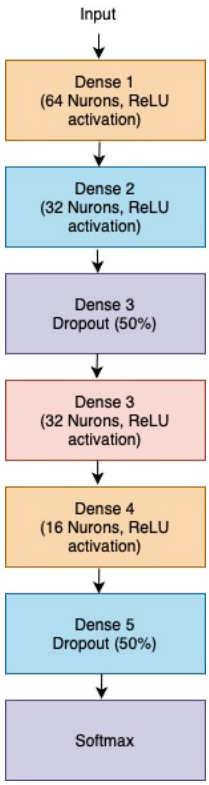

Fig. 4. Dense Network Design of Fusion Model – Task A

The implementation establishes a comprehensive pipeline for extracting and fusing features from multiple Convolutional Neural Network (CNN) models to train a unified classifier. It

begins by defining the extract features function, which isolates the penultimate layer of a given CNN model to capture high-level, discriminative features. This function creates a feature extractor model that takes inputs from the original CNN and outputs activations from the designated layer. The features are extracted for three separate models (model1, model2, and model3) using both training and testing datasets, resulting in feature representations (features_train and features_test) for each model. These features are concatenated along the sample dimension using NumPy's concatenate function to form fused feature sets (fused_train_features and fused_test_features), integrating diverse information learned by the individual models. Corresponding class labels are also combined to align with the fused datasets, while the total number of classes is updated to reflect the union of all unique classes from the three models.

Once the fused features are ready, a dense neural network as shown in the figure 4 is built to learn and predict from this combined feature representation. The architecture of the classifier includes an input layer matching the dimensionality of the fused features, followed by multiple dense layers with ReLU activations for non-linear learning. Dropout layers are incorporated at strategic points to mitigate overfitting by randomly disabling neurons during training. The network concludes with a softmax output layer that produces probabilities for each class in the combined class space. The classifier is compiled with the Adam optimizer for adaptive learning, sparse categorical cross-entropy loss for multi-class classification, and accuracy as a performance metric. This approach leverages the complementary strengths of multiple CNN models, enabling the unified model to make more robust and accurate predictions by harnessing the diversity of features learned by the individual networks.

D. Fusion/Merged Model – Task B

This design includes a workflow that extracts, fuses, and classifies features from multiple CNN models using a custom-designed dense network. The extract_features function isolates the penultimate layer of a CNN model, enabling it to capture high-level features that are passed through the model's pipeline. This process is applied to three distinct CNN models (model 1, model 2, and model 3) for both training and testing datasets, yielding feature representations for each dataset. These features are then fused using concatenation, creating comprehensive training and testing feature sets (fused_train_features and fused_test_features) that combine the strengths and complementary insights of the individual models. Corresponding class labels are similarly concatenated, ensuring alignment with the fused datasets. The total number of classes is calculated based on all unique labels across the three models, and the combined classes are documented for unified classification.

The fused features are passed into a dense network classifier with an enhanced architecture incorporating batch normalization, dropout layers, and L2 regularization for better generalization and robustness against overfitting. The model begins with a fully connected layer of 128 neurons followed by progressively smaller dense layers, each equipped with ReLU activation and regularization. Dropout layers are strategically placed to reduce dependency on specific neurons, and the model concludes with a softmax layer that predicts probabilities across the unified class space. The classifier is compiled with an Adam optimizer using a reduced learning rate for fine-tuning, and sparse categorical cross-entropy loss for multi-class classification. An early stopping mechanism is added to prevent overfitting and save the best weights based on validation loss. This approach effectively integrates and utilizes features from multiple CNNs to enhance classification performance while maintaining generalization.

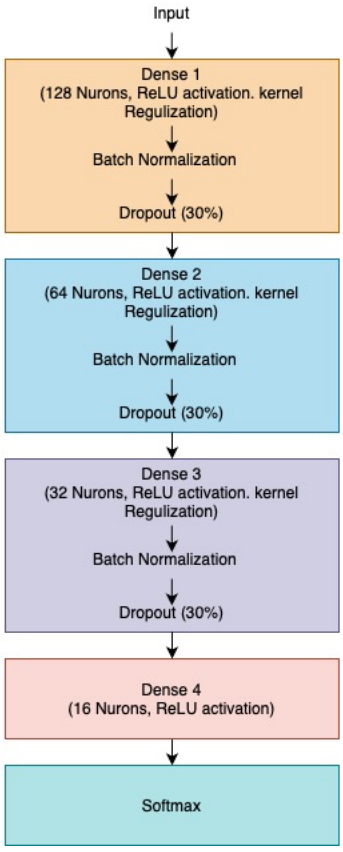

Fig. 5. Dense Network Design of Fusion Model – Task B

III. EVALUATION

A. TASK A - Performace

The classification report for the model 1 on testing data shows an overall accuracy of 82% across 500 test samples. Precision, recall, and F1-scores are well-balanced, with the macro and weighted averages at 0.82 for all metrics. Individual class performance highlights strong results for "apple" (F1-score: 0.87) and "dolphin" (F1-score: 0.90), while "bowl" has the lowest performance (F1-score: 0.70) due to lower recall. The model demonstrates good generalization but could benefit from improvements in handling challenging classes like "bowl."

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step
Classification Report for apple Model:
              precision    recall  f1-score   support

       apple       0.83      0.91      0.87       100
        bowl       0.76      0.65      0.70       100
       chair       0.88      0.82      0.85       100
     dolphin       0.88      0.92      0.90       100
        lamp       0.73      0.79      0.76       100

    accuracy                           0.82       500
   macro avg       0.82      0.82      0.82       500
weighted avg       0.82      0.82      0.82       500
```

Fig. 6. Classification Report of Model 1 – Task A

The classification report for the model 2 on testing data shows an overall accuracy of 84% across 500 test samples, with macro and weighted averages for precision, recall, and F1-score at 0.84. Among the classes, "aquarium_fish" achieves the highest performance with an F1-score of 0.89, indicating strong precision and recall. "Lawn_mower" also performs well, with an F1-score of 0.85. On the other hand, "chimpanzee" and "boy" have slightly lower F1-scores (0.82), and "elephant" shows moderate performance with an F1-score of 0.80. Overall, the model demonstrates consistent results but has room for slight improvements in handling specific classes like "elephant".

Classification Report for aquarium_fish Model:

	precision	recall	f1-score	support
aquarium_fish	0.90	0.89	0.89	100
boy	0.84	0.81	0.82	100
chimpanzee	0.84	0.80	0.82	100
elephant	0.79	0.80	0.80	100
lawn_mower	0.81	0.88	0.85	100
accuracy			0.84	500
macro avg	0.84	0.84	0.84	500
weighted avg	0.84	0.84	0.84	500

Fig. 7. Classification Report of Model 2 – Task A

The classification report for the model 3 on testing data indicates an overall accuracy of 77% across 500 test samples, with macro and weighted averages for precision, recall, and F1-score at 0.77. Among the classes, "bridge" and "leopard" achieve the highest performance, each with an F1-score of 0.84 and 0.83, respectively, demonstrating strong precision and recall. "Baby" shows a balanced performance with an F1-score of 0.74, while "flatfish" has the lowest F1-score of 0.68, indicating challenges in precision and recall. The model performs consistently but shows potential for improvement, particularly in the "flatfish" and "clock" categories.

Classification Report for baby Model:

	precision	recall	f1-score	support
baby	0.71	0.78	0.74	100
bridge	0.80	0.89	0.84	100
clock	0.83	0.69	0.75	100
flatfish	0.69	0.66	0.68	100
leopard	0.82	0.83	0.83	100
accuracy			0.77	500
macro avg	0.77	0.77	0.77	500
weighted avg	0.77	0.77	0.77	500

Fig. 8. Classification Report of Model 3 – Task A

Below table shows the performance of an individual models and fusion model of Task A.

TABLE I. TASK A PERFORMANCE

Model	Training Accuracy	Testing Accuracy
1	96%	81%
2	97%	83%
3	90%	76%
Fusion	84%	81%

B. TASK 3 - Performance

The classification report of model 1 on testing data reveals an overall accuracy of 74% for the model across 450 test samples. Among the five classes, "Hyena" and "Arctic_fox" achieve the highest F1-scores of 0.77 and 0.79, respectively, demonstrating strong recall. In contrast, "Chihuahua" shows the lowest F1-score of 0.69, indicating lower recall despite its high precision of 0.93. The macro and weighted averages for precision, recall, and F1-score hover around 0.73–0.76, highlighting reasonably balanced performance across the classes, with slight room for improvement in recall consistency.

	precision	recall	f1-score	support
Chihuahua	0.93	0.56	0.69	90
baboon	0.74	0.58	0.65	90
hyena	0.67	0.91	0.77	90
Arctic_fox	0.73	0.87	0.79	90
lynx	0.72	0.77	0.74	90
accuracy			0.74	450
macro avg	0.76	0.74	0.73	450
weighted avg	0.76	0.74	0.73	450

Fig. 9. Classification Report of Model 1 – Task B

The classification report of model 2 on testing data indicates an overall accuracy of 70% for the model on 450 test samples. The "African Hunting Dog" class achieves the highest F1-score of 0.77, driven by strong recall of 0.96, while "African Elephant" has the lowest F1-score of 0.57, despite perfect precision, due to its low recall of 0.40. The "Zebra" class also performs well with an F1-score of 0.85. The macro and weighted averages for precision, recall, and F1-score are consistent at around 0.69–0.77, reflecting moderate performance overall, with variations in recall and precision across different classes.

	precision	recall	f1-score	support
African Elephant	1.00	0.40	0.57	90
Hyena	0.53	0.87	0.66	90
Zebra	0.99	0.74	0.85	90
Patas	0.72	0.51	0.60	90
African Hunting Dog	0.64	0.96	0.77	90
accuracy			0.70	450
macro avg	0.77	0.70	0.69	450
weighted avg	0.77	0.70	0.69	450

Fig. 10. Classification Report of Model 2 – Task B

The classification report of model 3 on testing data shows an overall accuracy of 84% for the model on 100 test samples. The "African Hunting Dog" class performs slightly better with an F1-score of 0.85, supported by a high recall of 0.92. The "Collie" class achieves an F1-score of 0.83, with slightly higher precision at 0.90 but lower recall at 0.76. Both macro and weighted averages for precision, recall, and F1-score are consistent at 0.84–0.85, indicating balanced performance across the two classes.

	precision	recall	f1-score	support
Collie	0.90	0.76	0.83	50
African Hunting Dog	0.79	0.92	0.85	50
accuracy			0.84	100
macro avg	0.85	0.84	0.84	100
weighted avg	0.85	0.84	0.84	100

Fig. 11. Classification Report of Model 3 – Task B

Below table shows the performance of an individual models and fusion model of Task B.

TABLE II. TASK B PERFORMANCE

Model	Training Accuracy	Testing Accuracy
1	84%	74%
2	86%	70%
3	97%	86%
Fusion	91%	77%

IV. RELETED WORK

The related work is grounded in Convolutional Neural Network (CNN) research, which has transformed image classification by learning hierarchical features

Key CNN architectures include LeNet (early CNN for digit recognition), AlexNet (deeper with ReLU and dropout), VGG (uniform filters and depth), Inception (multi-scale feature extraction), ResNet (residual connections for deep networks), and MobileNet (efficient for mobile devices). CNNs are applied in domains like animal classification, medical imaging, and autonomous vehicles. This study uses these CNN fundamentals and explores feature-level fusion to improve classification.

V. CONCLUSION

The Project presented feature-level fusion model to enhance image classification. The experiments conducted in Task A and Task B highlight the potential of combining information from multiple CNN models, each specializing in a subset of classes. The results demonstrate that the fusion model can effectively leverage the complementary strengths of individual models, leading to improved classification accuracy and generalization. The proposed methodology provides a valuable framework for tackling complex image classification problems involving diverse or heterogeneous datasets, offering a robust solution to the limitations of single-model approach.