CSC 578 HW\#7 Intel Image Classification Competition (Fall 2023)

Name: Sadiya Amreen

Kaggle User Name: SADIYA AMREEN123

BEST MODEL : (KAGGLE Score : private : 0.97754 ; public: 1.01072) Model 12

For this homework , I tried constructing different Convolutional neural Network models by tuning number of hyperparameters and the best model out of my 27 versions submitted on Kaggle ,came out to be the of the version 7 (model 12) with the following specifications:

Number of filters : 16 , 32, 64 (layer-wise)
Padding : added
Strides : added
Additional layer added : 1
Batch normalization : added
Worked with all the optimizers (Adam, Adamax, Adagrad, Adadelta) : Adam worked best.

The approach I took for hyperparameters tuning:

1)Number of filters
2) Size of Filters
3)Padding
4)Strides
5)Additional layers until overfitting occurs
6)Working with regularizers L1, L2, L1L2
7)Dropout
8)Batch Normalization
9)data Augmentation

I sequentially tried all of the additions and it did not improve the CNN's performance.

Model 00:

Model Architecture:

Input Layer:
Rescaling layer: Scales input pixel values to a range of [0,1] by dividing them by 255.
Convolutional Layers:
Conv2D with 16 filters, each of size (3,3), and ReLU activation.
MaxPooling2D layer with a pool size of (2,2) to downsample the spatial dimensions.
Conv2D with 32 filters, each of size (3,3), and ReLU activation.
MaxPooling2D layer with a pool size of (2,2) for further downsampling.
Flatten Layer:
Flattens the output from the convolutional layers into a 1D array.
Dense Layers:
Dense layer with 128 neurons and ReLU activation.
Dense layer with 6 neurons (assuming it's a classification task with 6 classes) and softmax activation for multi-class classification.

Hyperparameters Experimented :

Initial model , did not experiment any hyperparameters.

```
loss: 0.7653 - accuracy: 0.7179 - val_loss: 0.7812 - val_accuracy: 0.7097
```

Model 01:

Model Architecture:

This model has an extra Conv2D layer with 16 filters and ReLU activation after the first max-pooling layer.

Hyperparameters Experimented:

I did not experiment with any hyperparameters in this model, because I wanted to see how will the CNN perform with an additional Convolutional layer alone.

Expectation:

The additional convolutional layer in model 01 increases the model's capacity to learn more complex hierarchical features from the input data. This might potentially lead to better performance.

Outcome:

Model 01 demonstrates an improvement in both training and validation accuracy compared to Model 00. Despite similar validation losses, the higher accuracy of Model 01 suggests that it has learned more effectively and generalized better to the validation set.

```
loss: 0.6261 - accuracy: 0.7688 - val_loss: 0.7796 - val_accuracy: 0.7361
```

Model 02:

Model Architecture:

This model has same architecture as Model 01, kept the additional Conv2D layer.

Hyperparameters Experimented:

Added 'padding' parameter with 'same' value to the Conv2D layers to include zero-padding, aiming to retain spatial dimensions after convolution.

Expectation:

Introducing padding is expected to mitigate information loss at the borders during convolution, potentially aiding in the preservation of spatial features.

Outcome:

The training accuracy is gradually increasing, indicating that the model is learning from the training data. However, the validation accuracy fluctuates, suggesting potential overfitting or sensitivity to the dataset. Keeping this Model for futher analysis.

```
loss: 0.6141 - accuracy: 0.7792 - val_loss: 0.7721 - val_accuracy: 0.7379
```

Model 03:

Model Architecture:

This model has same architecture as Model 02.

Hyperparameters Experimented:

Added 'strides' parameter with (2,2) values to the MaxPooling2D layers to increase the stride size during pooling.

Expectation:

Increasing the stride during pooling may reduce spatial dimensions faster, potentially extracting more

prominent features and reducing computational complexity.

Outcome:

The model exhibits a noticeable improvement in both training and validation accuracies compared to the previous Model 02. Training and validation losses are decreasing, suggesting better convergence and improved generalization. The stride modification in MaxPooling layers appears to have positively influenced the model's ability to capture relevant features. Keeping this Model for futher analysis.

```
loss: 0.6046 - accuracy: 0.7806 - val_loss: 0.6695 - val_accuracy: 0.7726
```

Model 04:

Model Architecture:

This model has same architecture as Model 02 .

Hyperparameters Experimented:

Added SpatialDropout2D layers with a dropout rate of 0.25 after each MaxPooling2D layer to introduce spatial dropout.

Expectation:

Spatial dropout is expected to improve generalization by randomly dropping entire feature maps during training, reducing overfitting and promoting more robust learning.

Outcome:

Model 03 outperforms Model 04, demonstrating higher training and validation accuracy and lower validation loss. Model 03 appears more effective in learning from the training data and generalizing to the validation set. Hence, removing Spatial dropouts for further analysis.

```
loss: 0.8376 - accuracy: 0.6866 - val_loss: 0.7528 - val_accuracy: 0.7354
```

Model 06:

Model Architecture:

This model has same architecture as Model 02 .Added Batch Normalization.

Hyperparameters Experimented:

Added BatchNormalization layers after each Conv2D layer to normalize the activations and stabilize training.

Expectation:

BatchNormalization is expected to accelerate training by normalizing activations and reducing internal covariate shift, potentially improving convergence and generalization.

Outcome:

Model 06 demonstrates significantly higher training accuracy but has a slightly lower validation accuracy compared to Model 04. The substantial difference between training and validation accuracies in Model 06 suggests potential overfitting. Will Explore regularization in further analysis.

```
loss: 0.4276 - accuracy: 0.8454 - val_loss: 0.8522 - val_accuracy: 0.7215
```

Optimizers:

Adamax (Model 07):

Expectation: Anticipated improved performance due to the adaptive learning rate behavior of Adamax.
Observation: Confirmed, as Model 07 achieved the highest validation accuracy (75.08%).

Adagrad (Model 08):

Expectation: Initial expectation was a competitive performance.
Observation: Achieved a good accuracy but was outperformed by Adamax.

Adadelta (Model 09):

Expectation: Expected Adadelta to excel given its adaptive learning rate behavior and robustness.
Observation: Contrary to expectations, Model 09 performed poorly, suggesting that other optimizers like Adamax might be more suitable for the given task.

Model 07 with the Adamax optimizer demonstrates the best balance of training and validation accuracy among the three optimizers considered. Adadelta's performance did not meet expectations, indicating the importance of empirical testing to determine the most suitable optimizer for a specific task.

Dropout Expectations and Observations:

Model 10 (Adamax with Dropouts):

Expectation: Anticipated that dropouts would enhance generalization and prevent overfitting leading to improved performance.
Observation: Confirmed, as Model 10 achieved a good validation accuracy (75.76%).

Model 11 (Adam with Dropouts):

Expectation: Expected improved generalization due to the introduction of dropout layers.
Observation: Contrary to expectations, Model 11 exhibited poor performance, especially on t he validation set, indicating potential overfitting.

Despite the expectation that dropout layers would contribute to improved generalization, the a ctual performance varied between models. Model 10, with the Adamax optimizer and dropouts outperformed Model 11, which used Adam with dropouts.

Best model:
Model 12: Expectation, Outcome, Hyperparameters, and Architecture

Expectation:
Expected higher accuracy due to the increased model complexity and higher filter counts

Outcome:
Training Accuracy: Achieved a high training accuracy of 86.90%.
Validation Accuracy: Attained a decent validation accuracy of 74.33%.

Observations: The model exhibited good generalization, considering the increased complexity, with no signs of severe overfitting.

Hyperparameters Experimented:

Optimizer: Adam with default learning rate (0.001).

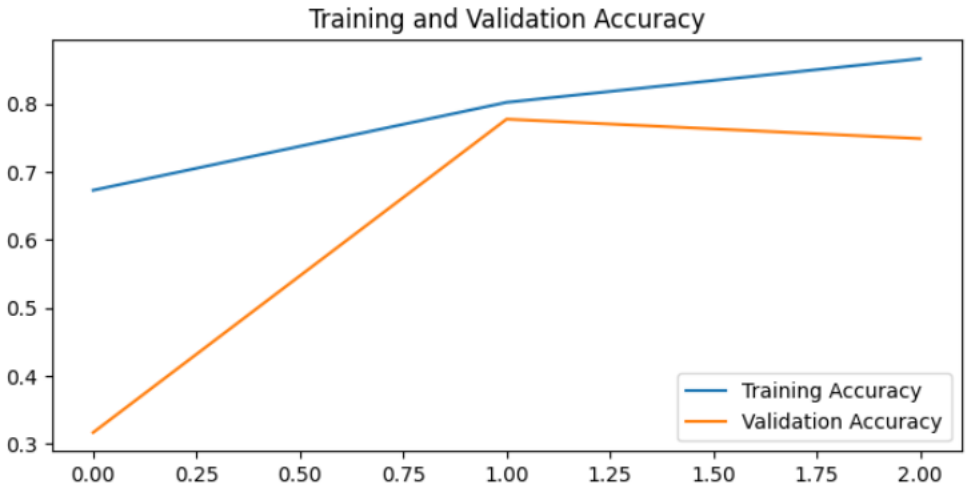
Filter Counts: Increased filter counts to 16, 32, and 64 in successive convolutional layers.

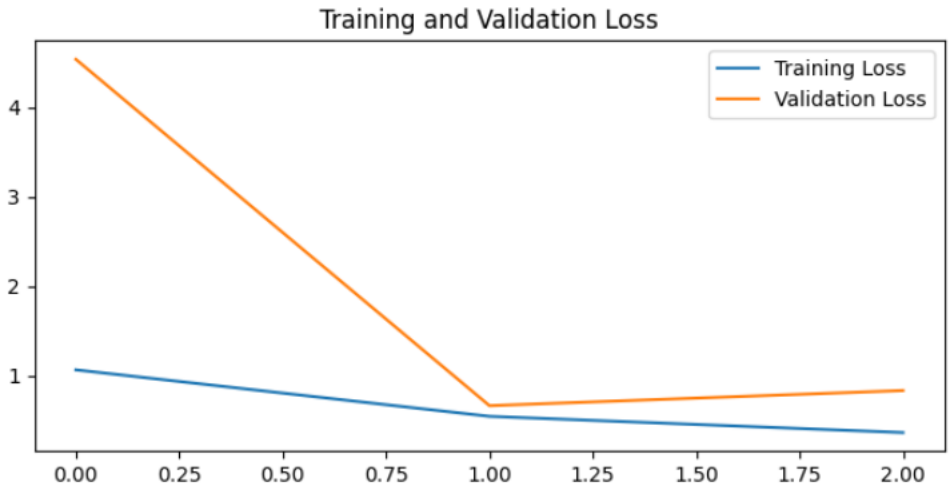
Batch Normalization: Applied batch normalization after each convolutional layer.

Flatten and Dense Layers: A single dense layer with 128 neurons before the output layer.

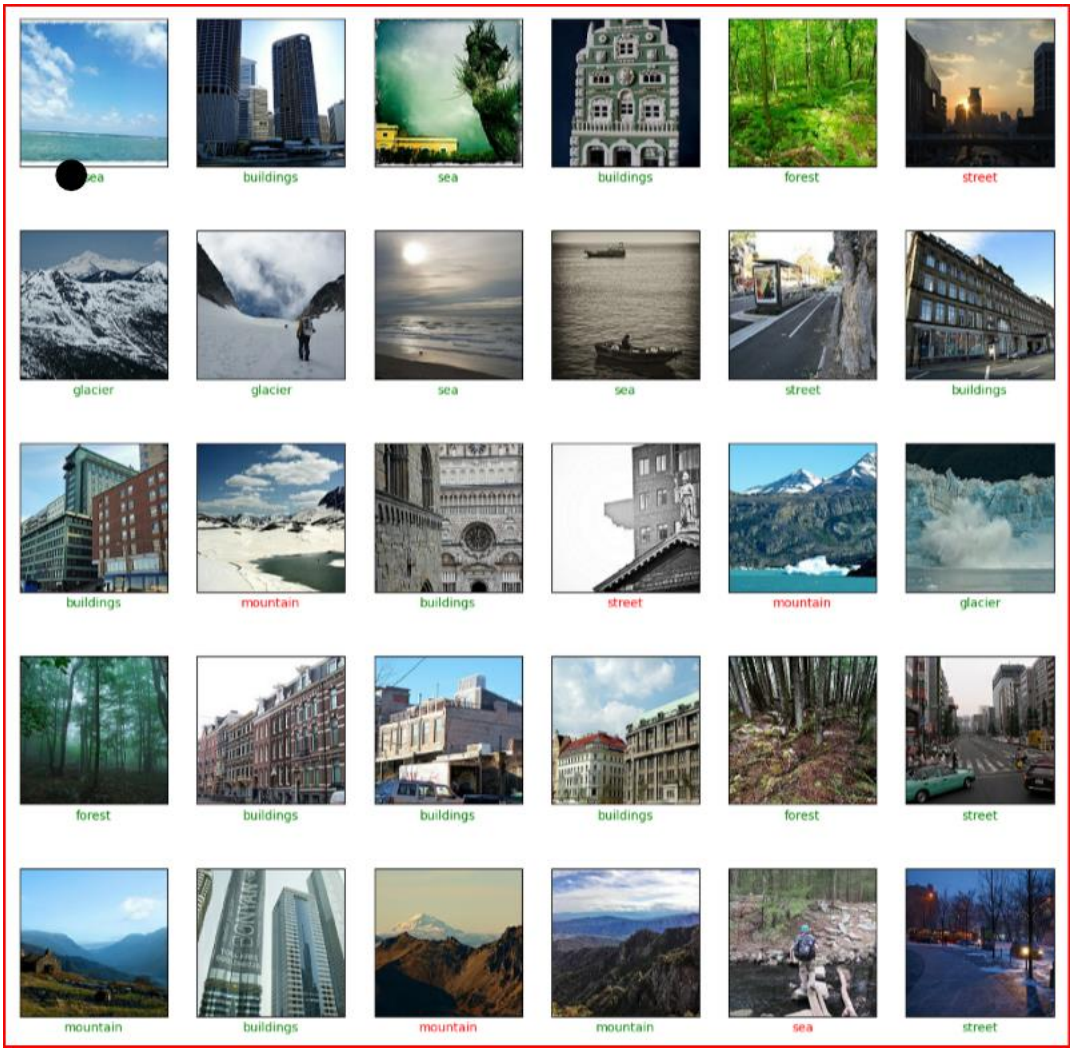
Model 12 demonstrated a balanced performance with both high training and validation accuracies.

The increased complexity, achieved by adding more layers and higher filter counts, contributed positively to the model's ability to generalize.





loss: 0.3657 - accuracy: 0.8664 - val_loss: 0.8355 - val_accuracy: 0.7490



Only 5 predictions went wrong for my best model.

Model Refinement Details - Experiments Leading to Model 12

Regularization Techniques (L1, L2, L1L2): (Model 14 -18)

Expectation: Introducing regularization to prevent overfitting and improve model generalization.
Outcome: The addition of L1, L2, and L1L2 regularizers did not significantly improve performance.

Non Best Model :
The impact of L1, L2, and L1L2 regularization on our models, aiming to prevent overfitting,

Unfortunately, the expected performance boost did not materialize. These models didn't show a clear advantage despite theoretical benefits."

Filter Size Adjustment:

Expectation: Changing filter sizes to capture different levels of abstraction and improve feature extraction.
Outcome: Altering filter sizes did not yield a notable improvement in performance.

Additional Layers:

Expectation: Increasing model complexity by adding more convolutional layers.
Outcome: While this improved model complexity, it did not lead to a significant boost in accuracy.

Data Augmentation:

Expectation: Augmenting the dataset to increase diversity and improve model robustness.
Outcome: Data augmentation, although a good practice, did not result in a substantial performance gain.

Model	Architecture and Hyperparameters	Outcome
00	Rescaling > Conv2D(16) > MaxPooling2D > Conv2D(32) > MaxPooling2D > Flatten > Dense(128) > Dense(6)	loss: 0.7653, accuracy: 0.7179, val_loss: 0.7812, val_accuracy: 0.7097

01	Additional layer added after maxpooling 1	loss: 0.6261, accuracy: 0.7688, val_loss: 0.7796, val_accuracy: 0.7361
02	Padding added	loss: 0.6141, accuracy: 0.7792, val_loss: 0.7721, val_accuracy: 0.7379
03	Strides added	loss: 0.6046, accuracy: 0.7806, val_loss: 0.6695, val_accuracy: 0.7726
04	Rescaling > Conv2D(16) > MaxPooling2D > SpatialDropout2D > Conv2D(16) > MaxPooling2D > SpatialDropout2D > Conv2D(32) > MaxPooling2D > SpatialDropout2D > Flatten > Dense(128) > Dense(6)	loss: 0.8376, accuracy: 0.6866, val_loss: 0.7528, val_accuracy: 0.7354
05	Spatial dropout added to one layer	loss: 0.6772, accuracy: 0.7471, val_loss: 0.7194, val_accuracy: 0.7412
06	Batch Normalization added to all the layers	loss: 0.4276, accuracy: 0.8454, val_loss: 0.8522, val_accuracy: 0.7215
07	Adamax optimizer	loss: 0.2823, accuracy: 0.9061, val_loss: 0.7795, val_accuracy: 0.7508
08	Adagrad optimizer	loss: 0.5896, accuracy: 0.7928, val_loss: 0.7589, val_accuracy: 0.7233
09	adadelta	loss: 1.2968, accuracy: 0.5052, val_loss: 1.2204, val_accuracy: 0.5406
10	Adamax and dropout in last layers	loss: 0.5289, accuracy: 0.8052, val_loss: 0.6966, val_accuracy: 0.7576
11	Adam and dropout in last layers	loss: 0.6318, accuracy: 0.7675, val_loss: 5.7331, val_accuracy: 0.4226
12	Changed number of filters	**loss: 0.4096, accuracy: 0.8490, val_loss: 0.7880, val_accuracy: 0.7533**

FINAL Conclusion and Best Model (Model 12):

Observation: Despite various attempts and iterations, the best-performing model was Model 12.

Factors Leading to Selection:

Filter Counts: Gradually increased from 16 to 32 and 64.
Batch Normalization: Applied after each convolutional layer.
Overall Complexity: Achieved a balance between model complexity and generalization.
Justification: Model 12 demonstrated the highest validation accuracy (75.33%) among the experimented architectures.

Reactions and Reflections:

The assignment presented a formidable challenge, demanding considerable time and effort, leading to a rigorous and often stressful experience. The complexity of the task required substantial work outside of regular class hours. While the assignment provided a deep dive into various concepts, the intensity of juggling multiple subjects simultaneously was notable. In hindsight, a preference for a more focused approach emerged, where tackling one subject per quarter could have allowed for a more in-depth exploration and a more manageable workload. Despite the challenges, the assignment proved to be a valuable learning experience, significantly enhancing my knowledge base.