IBM Report for Deep Learning Project

Code on GitHub

Data Processing:
https://github.com/luisresende13/octa-flood-images/blob/main/Get%20Data%20from%20Videos%20Collection.ipynb

Data Sampling:
https://github.com/luisresende13/octa-flood-images/blob/main/Example%20Split%20%2B%20Copy%20Images%20into%20Split%20Folders.ipynb

Model Training and Evaluation
https://github.com/luisresende13/ibm-machine-learning-specialization/blob/main/{
MISSING
}.ipynb

Main Objective of the Analysis

The main goal of this analysis is to build a model that predicts the presence of flood events in images from street cameras in urban scenarios, by applying an image classification neural network.

The target variable for the model was a binary flag, signifying the presence of a significant flood event. By using convolutional neural networks, visual features can be extracted from the images to be used as predictors. The aim is to build a model capable of identifying the presence of such events based on these features.

Brief Description of the Data Set and Summary of Its Attributes

There was two main pieces of raw data used in the exercise:

1. A bucket in Google Cloud Storage containing the actual recorded videos from street cameras in Rio de Janeiro, as mp4 files.
2. A mongo collection with a set of records corresponding to the videos in the bucket, which includes fields such as the code of the camera, the initial and final timestamps, whether the video passed through the labeling process and a multi-class field with the labels of the video, if any. The videos are labeled according to the gravity of the flood event, if any. The videos are labeled as one or more of the following:
    a. poça (pud)
    b. lâmina (water blade)
    c. bolsão (water bag)
    d. alagamento (flood)

This data was collected by a startup called Octa City as part of a project in partnership with the Control and Operations Center of Rio de Janeiro.

Brief Summary of Data Exploration and Actions Taken for Data Cleaning and Feature Engineering

The two pieces of raw data were processed into another three artifacts. These are:

1. Images: Videos in the bucket were downloaded and broken into jpeg images.

2. Videos dataset: The mongo collection was downloaded and converted to a pandas dataframe.
3. Images dataset: The videos dataset and extracted images were used to construct a dataset representing all available images and associated labels.

The target variable was created by applying binarization to the original categories. The ‘pud’ and ‘water blade’ categories were replaced by ‘normal’ and ‘water bag’ and ‘flood’ by ’flood’. Videos that passed through labeling but with no label were also labeled as ‘normal’ for classification.

Videos dataset:

- Video files in dataset (rows): 62017
- Total megabytes (MBs): 78.405
- Time to download: 13.6
- Time to save dataframe: 0.9
- Time Total: 14.7 s
- Videos watched: 4414
- Videos labeled: 1776
- Cameras with labels: 109
- Watched videos count per label:
    - normal: 2669
    - poça: 1348
    - lâmina: 213
    - bolsão: 94
    - alagamento: 36

Only videos marked as “watched” were downloaded and broken into images since the remaining videos might contain unlabeled flood events.

Downloaded videos:

- Video files: 4414

Images extracted and images dataset:

- Images: 168494
- Images watched: 168494 / 168494
- Images count per label:
    - normal: 100031
    - poça: 53218
    - lâmina: 7979
    - bolsão: 3612
    - alagamento: 1417
- Images count per label of the *target variable*:
    - normal 153249
    - flood 13008

Data Sampling

Firstly, random undersampling was applied between the samples belonging to “water blade”, “water bag” and “flood” classes. The “flood” class was the least represented, having 1417 samples. The other two classes were sampled using the same sample size, there is 1417 samples, resulting in a balance between these three classes and a dataset with the following distributions:

- Image tags:

    - normal   100031
    - poça   53218
    - alagamento   1417
    - bolsão   1417
    - lâmina   1417

- Image flood labels:
    - 0   153249
    - 1   4251

Subsequently, random undersampling was applied between the flood classes (0 and 1):

- Image flood labels:
    - 0   4251
    - 1   4251

Finally, uniform random sampling was used to reduce the dataset to 1000 samples (500 per class), in order to reduce computation time and also repetitive frames.

- Image flood labels:
    - 0   500
    - 1   500

The code of the camera was used as the grouping factor when applying the "Group K Fold" split using the StratifiedGroupKFold method from “sklearn”. The split method was chosen to create the train, test and validation sets without any camera in common in between them.

The method was applied two times, one with K=9 folders, to select the test folder, and one with K=8 folders done on the training set selected in the previous step, to separate the training and validation sets. Using both the test and validation sets will show whether the model will be able to generalize for images from unseen cameras.

The split created a training set (791 samples - 79.1% and 118 cameras), a testing set (114 samples - 11.4% and 9 cameras) and a validation set (95 samples - 9.5% and 13 cameras), all maintaining the same ration (~50%) across classes (0 and 1) than before splitting.

Group K Fold split sets:

```
Groups intersecting train and test sets: 0
Groups intersecting train and val sets: 0
Groups intersecting test and val sets: 0

      cameras  images  0  1       0 (%)   1 (%)   class ratio (%)
train   118.0   791.0   394.0   397.0   78.8    79.4    49.81
test    9.0     114.0   58.0    56.0    11.6    11.2    50.88
val     13.0    95.0    48.0    47.0    9.6     9.4     50.53
```

Summary of Training Three Different Image Classification Neural Networks

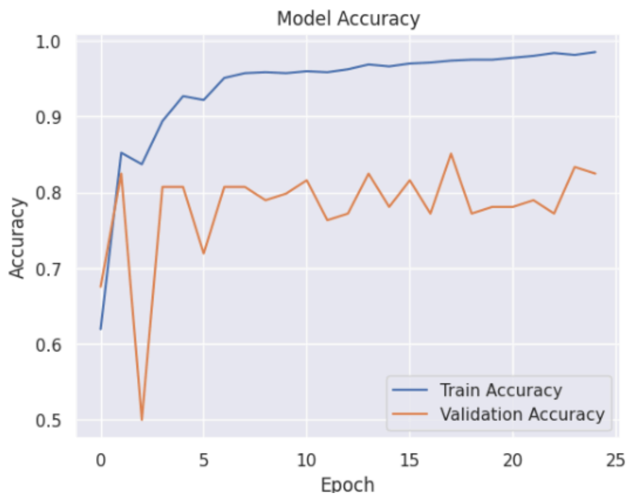
The performance of three deep learning models trained for binary classification were evaluated. The models are a Simple Convolutional Neural Network (CNN), MobileNetV2, and a CNN with Dropout.

The evaluation was done on training, test, and validation sets, using standard metrics such as precision, recall, and F1-score.

All models trained over 25 epochs using batches of 32 labeled images.

Initially used a default learning rate, but all models faced challenges of fast convergence and oscillations in loss and accuracy. The solution was to reduce the learning rate iteratively during training and evaluating the performance visually.

Kaggle's multi GPU cloud infrastructure was used in order to accelerate training time.



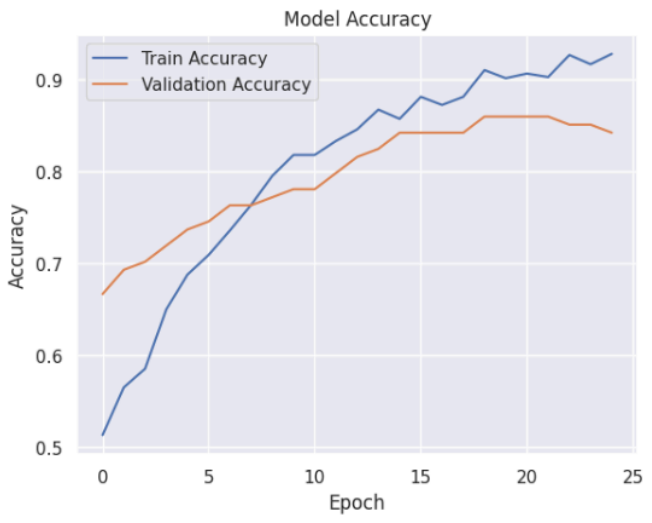
Model 1: Simple Convolutional Neural Network (CNN)

Optimal Learning Rate: 0.000002

Training Set: Achieved 99%, with a minimal misclassification rate as shown by the confusion matrix and classification report in the next section.

Test Set: Medium performance with an accuracy of 85%. This highlights the model's ability to generalize to new data.

Validation Set: Similar performance of the validation set to the test set with an accuracy of 88%. It demonstrates consistent behavior of the model on different cameras, since the train, test and validation sets each contain a different set of cameras.



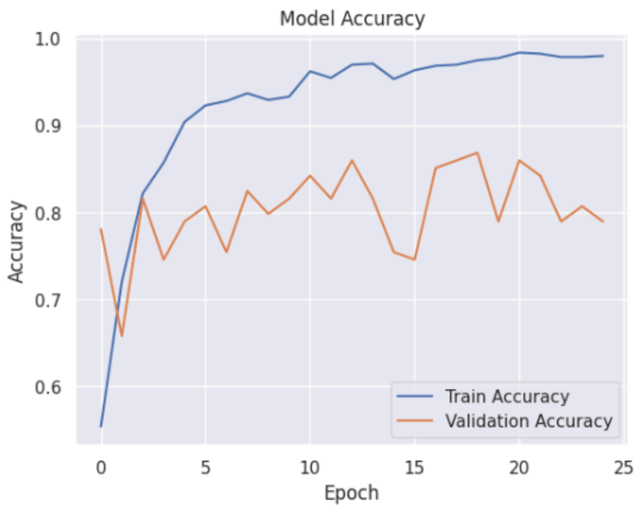
Model 2: MobileNetV2

Optimal Learning Rate: 0.000002

Training Set: Accuracy of 81% on the training set. Precision and recall show a balanced performance for positive and negative samples.

Test Set: Generalized well to the test set with an accuracy of 82%, and a balanced precision-recall trade-off.

Validation Set: Low generalization on the validation set, which indicates a potential area for improvement with hyperparameter tuning.



Model 3: CNN with Dropout

Optimal Learning Rate: 0.00001

Training Set: Accuracy of 99% on the training set, comparable to Model 1. Similar to Model 1, precision, recall, and F1-score values indicate robust performance.

Test Set: Accuracy of 85%, which indicates good generalization in handling new cameras.

Validation Set: Accuracy of 85%, which also shows the model's capability to generalize for unseen cameras.

Note: The values from the charts above are not necessarily from the same training execution of the evaluation results presented in this report and serve for illustrative purposes.

Evaluation Results

These reports demonstrate how well each model performed on the training, test, and validation sets.

Model 1 - Simple Convolutional Neural Network (CNN):

- Training Set:
  - Confusion Matrix:
  -
  - [[389   5]
  - [1 396]]
  -
  - Classification Report:
  -
  -             precision   recall f1-score   support
  -
  -          0   1.00   0.99   0.99   394
  -          1   0.99   1.00   0.99   397
  -
  -   accuracy   0.99   791
  - macro avg   0.99   0.99   0.99   791

- Test Set:
  - Confusion Matrix:
  -
  - [[48 10]
  - [7 49]]

  - Classification Report:

-

-
  ```
              precision    recall  f1-score   support
  ```
-

-
  ```
           0       0.87      0.83      0.85        58
  ```

-
  ```
           1       0.83      0.88      0.85        56
  ```
-

-
  ```
  accuracy                          0.85       114
  ```
- macro avg 0.85 0.85 0.85 114

- Validation Set:
    - Confusion Matrix:
    -
    - [[43 5]
    - [6 41]]

    - Classification Report:
    -

    -
      ```
                  precision    recall  f1-score   support
      ```
    -

    -
      ```
               0       0.88      0.90      0.89        48
      ```

    -
      ```
               1       0.89      0.87      0.88        47
      ```
    -

    -
      ```
      accuracy                          0.88        95
      ```
    - macro avg 0.88 0.88 0.88 95

Model 2 - MobileNetV2:

- Training Set:
    - - Confusion Matrix:
    -
    - [[337 57]
    - [97 300]]

    - Classification Report:
    -

    -
      ```
                  precision    recall  f1-score   support
      ```
    -

    -
      ```
               0       0.78      0.86      0.81       394
      ```

    -
      ```
               1       0.84      0.76      0.80       397
      ```
    -

    -
      ```
      accuracy                          0.81       791
      ```
    - macro avg 0.81 0.81 0.80 791

- Test Set:
    - Confusion Matrix:
    -
    - [[52 6]
    - [15 41]]

    - Classification Report:
    -
    - precision recall f1-score support
    -
    - 0 0.78 0.90 0.83 58
    - 1 0.87 0.73 0.80 56
    -
    - accuracy 0.82 114
    - macro avg 0.82 0.81 0.81 114

- Validation Set:
    - Confusion Matrix:
    -
    - [[45 3]
    - [34 13]]

    - Classification Report:
    -
    - precision recall f1-score support
    -
    - 0 0.57 0.94 0.71 48
    - 1 0.81 0.28 0.41 47
    -
    - accuracy 0.61 95
    - macro avg 0.69 0.61 0.56 95

Model 3 (CNN with Dropout):

- Training Set:
    - Confusion Matrix:
    -
    - [[389 5]
    - [1 396]]

    - Classification Report:
    -
    - precision recall f1-score support
    -
    - 0 1.00 0.99 0.99 394
    - 1 0.99 1.00 0.99 397
    -
    - accuracy 0.99 791
    - macro avg 0.99 0.99 0.99 791

- Test Set:
    - Confusion Matrix:
    -
    - [[48 10]
    - [7 49]]

    - Classification Report:
    -
    -           precision   recall f1-score   support
    -
    -        0   0.87   0.83   0.85   58
    -        1   0.83   0.88   0.85   56
    -
    - accuracy   0.85   114
    - macro avg 0.85 0.85 0.85 114

- Validation Set:

    - Confusion Matrix:
    -
    - [[35 13]
    - [0 47]]

    - Classification Report:
    -
    -              precision   recall f1-score   support
    -
    -           0   1.00   0.73   0.84   48
    -           1   0.78   1.00   0.88   47
    -
    -     accuracy   0.86   95
    -    macro avg   0.89   0.86   0.86   95

Recommended Final Model

After evaluation of the three image classification models, the CNN with Dropout is recommended as the final model.

Trade-off: Good balance in accuracy between classes and good improved resistance to overfitting.

Convergence: Successfully optimized learning rate, achieving a stable convergence of the model.

The model's balance between accuracy, convergence, and resilience to overfitting makes it an optimal choice for deployment in real-world applications.

Key Findings and Insights

The trained neural networks achieved a fair accuracy in classifying the presence of flood events in urban scenes captured by street cameras. The models exhibited medium performance across various weather conditions, lighting scenarios, and camera perspectives, with potential for real-world deployment.

Real-time Inference: Finding balance between accuracy and inference speed was crucial for fitness to practical deployment.

GPU usage: Kaggle's GPU infrastructure accelerated training time, allowing for more experiments and hyperparameter tuning.

Learning rate challenge: The learning rate reduction significantly improved stability and convergence in the training process across all models.

Final model insights:

- Regularization: Implemented dropout layers to mitigate overfitting, resulting in improved generalization.

- Precision and Recall: Good balance between precision and recall for both classes.

- Validation Performance: Accuracy of 85% balanced across classes, which shows the model's capability to generalize for unseen cameras.

Next Steps

Hyperparameter Tuning:

- Optimization using hyperparameter tuning for specific use-cases.

Ensemble Approaches:

- Ensemble models for potential increase in performance.

Data Augmentation:

  - Diversification: Enhancing the model's ability to generalize to unseen data.
  - Methods: Rotation, scaling, and flipping.

Transfer Learning:

  - Extended Pre-training: Use longer periods for training, to allow the model to capture more complex features from the pre-trained weights.