AI 5100 HACKATHON

Presentation

Ritvik Sai Chippa - CS21BTECH11054
Nishanth Bhoomi - CS21BTECH11040

Problem Statement

Challenge:

The aim is to complete a standard multiclass classification task with the highest accuracy on both public and private leaderboards.

Dataset:

- Training folder contains 50 sub-folders, each representing a class.
- Test folder contains images for prediction.

Constraints:

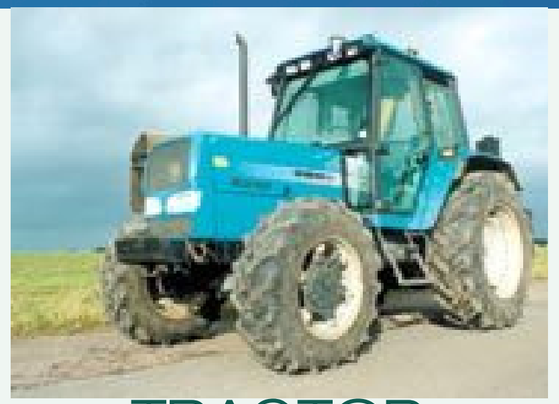
- No pretrained models allowed; train models from scratch.
- Limited to using PyTorch (`torch.nn.*` and `torch.nn.functional.*`) and custom implementations.
- Pretraining permitted if implemented by students themselves.

Objective:

Optimize a model to achieve maximum accuracy while adhering to specified constraints.

EXAMPLE IMAGES

DOG



TRACTOR

CANDLE

LOBSTER

JELLYFISH

BEE

HOUSE

BUS

PIZZA

COFFEE

BUTTERFLY

PENGUIN

INSIGHTS

- The dataset includes around 1,300 photos per class spread over 50 classes, making for a varied collection of photographs.
- Images' initial sizes vary; however, for model consistency, they are standardised to 224 by 224 pixels. Testing is done on 38000 images.
- To help in model learning, normalization is used to the training dataset to scale pixel values.
- Recognized that the train and test datasets may have different distributions, emphasizing the significance of both model complexity selection and generalization.
- Utilized the Adam optimizer and Cross Entropy Loss for training models.

STRATEGY 1:

INITIAL ATTEMPT WITH RESNET50

- **Model Architecture: ResNet50 without pretrained weights.**
- **Parameters: Approximately 25.5 million trainable parameters.**
- **Loss Function: Cross Entropy Loss.**
- **Optimizer: Adam optimizer.**
- **Hyperparameters: Learning rate, batch size, number of epochs.**
- **Regularizers: None.**
- **Data Preprocessing: Resized images to 224x224, normalization.**
- **Training: 10 epochs.**
- **Result: Achieved 39% accuracy.**

Analysis:

- **Worked: ResNet50 provided a solid baseline for the classification task. It effectively learned features from the images and made predictions.**
- **Failed: Despite being a powerful architecture, the model struggled to achieve high accuracy, possibly due to the lack of pretrained weights and limited training data.**

STRATEGY 2:

ATTEMPT WITH CNN

- **Model Architecture: CNN.**
- **Parameters: Trainable parameters within CNN.**
- **Parameter Count: Approximately 1.2 million parameters.**
- **Loss Function: CrossEntropyLoss.**
- **Optimizer: Adam optimizer.**
- **Hyperparameters: Learning rate, batch size, number of epochs.**
- **Regularizers: None.**
- **Data Preprocessing: Resized images to 64x64, normalization.**
- **Training: 10 epochs.**
- **Result: Achieved 43% accuracy.**

Analysis:

- **Worked: The CNN architecture showed improved performance compared to the initial ResNet50 attempt. It provided more flexibility in designing the network architecture tailored to the dataset.**
- **Failed: While achieving better accuracy than ResNet50, the CNN still fell short of achieving high accuracy, indicating the need for further optimization.**

STRATEGY 3:

VISION TRANSFORMER MODEL

- **Model Architecture: Vision Transformer (ViT).**
- **Parameters: Approximately 86 million trainable parameters within ViT.**
- **Loss Function: Cross Entropy Loss.**
- **Optimizer: Adam optimizer.**
- **Hyperparameters: Learning rate, batch size, number of epochs.**
- **Regularizers: None.**
- **Data Preprocessing: Resized images to 224x224, random flip and rotation, normalization.**
- **Training: 1 epoch due to longer training time.**
- **Result: Accuracy not known.**

Analysis:

- **Worked: The Vision Transformer (ViT) model demonstrated promising potential, leveraging self-attention mechanisms for image classification.**
- **Failed: Because of the increased training time each epoch, just one epoch was run, and the accuracy was unknown. Limited training data and resources prevented a an evaluation of ViT's performance. .**

STRATEGY 4:

IMPROVED CNN MODEL

- **Model Architecture: CNN.**
- **Parameters: Trainable parameters within CNN.**
- **Parameter Count: Approximately 1.2 million parameters.**
- **Loss Function: CrossEntropyLoss.**
- **Optimizer: Adam optimizer.**
- **Hyperparameters: Learning rate, batch size, number of epochs.**
- **Regularizers: None.**
- **Data Preprocessing: Resized images to 64x64, normalization.**
- **Training: 20 epochs.**
- **Result: Achieved 45% accuracy.**

Analysis:

- **Worked: The CNN model showed incremental improvement in accuracy compared to previous strategies. Increasing the number of training epochs also contributed to better performance.**
- **Failed: Despite the improvements, the model still fell short of achieving high accuracy, suggesting the need for further refinement in architecture or training strategy.**

STRATEGY 5:

DENSENET121 MODEL

- **Model Architecture: DenseNet121 without pretrained weights.**
- **Parameters: Approximately 6.9 million trainable parameters within DenseNet121**
- **Loss Function: Cross Entropy Loss.**
- **Optimizer: Adam optimizer.**
- **Hyperparameters: Learning rate, batch size, number of epochs.**
- **Regularizers: None.**
- **Data Preprocessing: Resized images to 224x224, normalization.**
- **Training: 10 epochs.**
- **Result: Achieved 49% accuracy.**

Analysis:

- **Worked: DenseNet121 emerged as the best-performing strategy, achieving the highest accuracy among all attempts. Its deep architecture and dense connectivity effectively captured intricate features from the images.**
- **Failed: While DenseNet121 yielded the highest accuracy, there is still room for improvement. Fine-tuning hyperparameters and exploring advanced techniques could potentially enhance performance further.**

CONCLUSION

Why It Worked:

- DenseNet121 emerged as the best-performing method, with the highest accuracy of all attempts.
- Its deep architecture and dense connectivity efficiently captured complex characteristics in the photos.
- DenseNet's dense connections enabled feature reuse and gradient flow throughout the network, resulting in increased feature extraction and learning.

What Could've Been Done: –

- Implementation of More Models: Implementing other models, such as ConvNextTiny or ensemble models, to experiment with different architectures and ensemble learning methods.
- Giving More Time to ViT Models : The ViT model could have given more training time in order to realise its capabilities. Due to the longer training period each epoch, just one epoch was run, limiting the ability to evaluate its performance.

What fell short:–

- Limited model exploration: More testing using different architectures could have given deeper understanding into appropriate model design for the task.
- Optimisation Challenges: The CNN model fell short of reaching high accuracy. This shows that additional optimisation of architecture design, hyperparameters, and training methods may be needed to increase performance.