Exploring Vision-Based Models for Land-Usage Classification Using Remote Sensing Imagery Data

Presented By - Group 163
Anirudh Kaluri (akaluri)
Rishi Singhal (rsingha4)
Yazhuo Gao (ygao46)

Motivation - Why Land Use Classification?

- Understanding land-use patterns is crucial for sustainable resource management, urban planning, and environmental conservation.

- When combined with deep learning, these images offer insights for disaster recovery, resource allocation, precision agriculture, biodiversity monitoring, and infrastructure planning. A core challenge is accurately classifying land-use patterns from satellite imagery.

Problem Statement

Enhance the understanding of deep learning (DL) models' usefulness for land-use classification using three vision-based neural networks to classify remote sensing images

Dataset Used

- To explore the task of land-use classification, we use the UC-Merced Land-Use Dataset available at Kaggle [1].

- It contains satellite images of different urban regions in the US extracted from USGS National Map Urban Area Imagery Collection.
- 21,00 RGB satellite images, most of size 256x256 pixels, across 21 land-use classes.

Dataset Used

Input & Output of the Task

- *Input:*

  A RGB satellite image representing a region of land use, e.g., a forest.

- *Output:*

  A predicted class label (e.g., forest, river) corresponding to the land-use image feeded to the network.

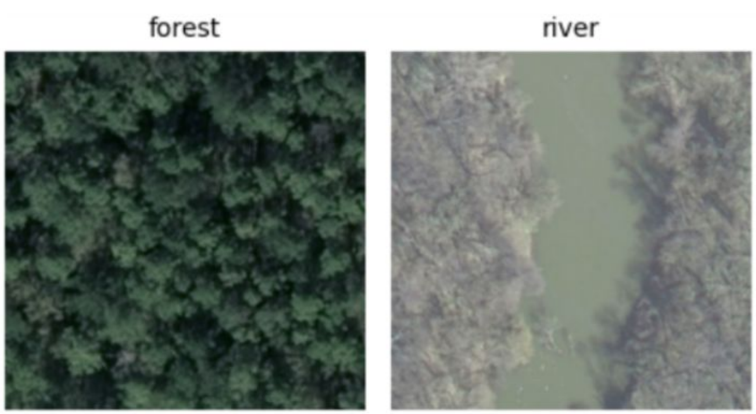


Image Categories

0: agricultural
1: airplane
2: baseballdiamond
3: beach
4: buildings
5: chaparral
6: denseresidential
7: forest
8: freeway
9: golfcourse
10: harbor
11: intersection
12: mediumresidential
13: mobilehomepark
14: overpass
15: parkinglot
16: river
17: runway
18: sparseresidential
19: storagetanks
20: tenniscourt

Prior Work

- Rishi has previously worked with vision based models for different classification tasks.

Methodology

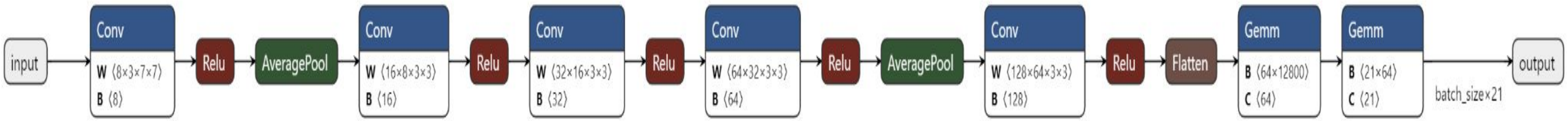
Data Splitting & Preprocessing

- *Data splitting* - We create a DataLoader which splits train set, validation set and test set - as with 70%, 15% and 15%.
- *Data Resize and normalization* - We resize all images to 256x256 and normalize to 0 mean & 1 std dev to maintain consistency.
- *Transformations* - We try multiple combinations of transformations to ensure better training performance.

```
Transform_pipeline = transforms.Compose([
    transforms.Resize((256, 256)),  # Resize images
    transforms.RandomHorizontalFlip(p=0.5),  # Flip images horizontally with 50% probability (74%)
    transforms.RandomRotation(degrees=30),  # Rotate images randomly within ±30 degrees (74%)
    # transforms.RandomResizedCrop(256, scale=(0.8, 1.0)),  # Crop images randomly (68%)
    # transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),  # Adjust brightness, contrast, etc. (65%)
    # transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),  # Apply random translation (72%)
    # transforms.RandomPerspective(distortion_scale=0.2, p=0.5),  # Apply random perspective transformations (69%)
    transforms.ToTensor(),  # Convert image to tensor
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])  # Normalize images
])
```

Baseline CNN Model

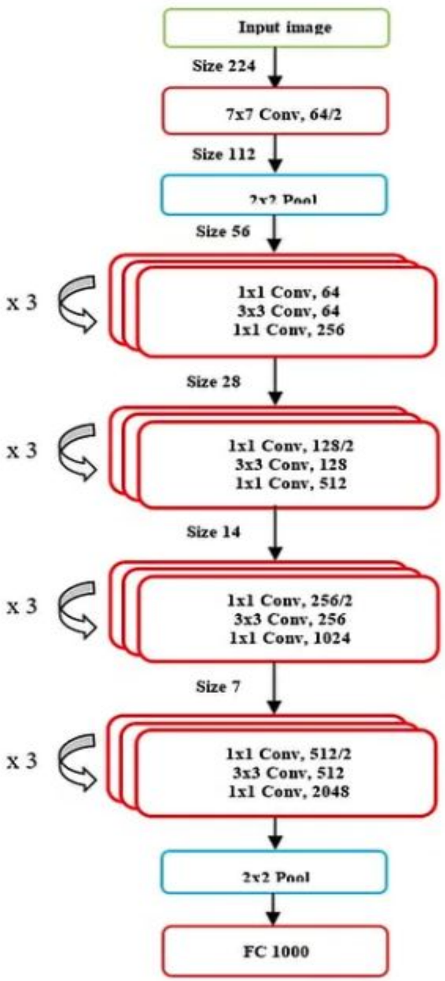
- *CNN Model* - We use a CNN consists of 5 2D-convolution layers with BatchNorm, ReLU, and AvgPool. A classifier with dropout is followed at last to handle the features and output (21x1 size). Below is the architecture diagram:



- *Hyperparameters*

Dropout Rate	Learning Rate	Weight Decay	Epochs	Loss function	Optimizer
p = 0.3	3e-4	1e-4	100	Cross Entropy Loss	SGD/Adam

ResNet50



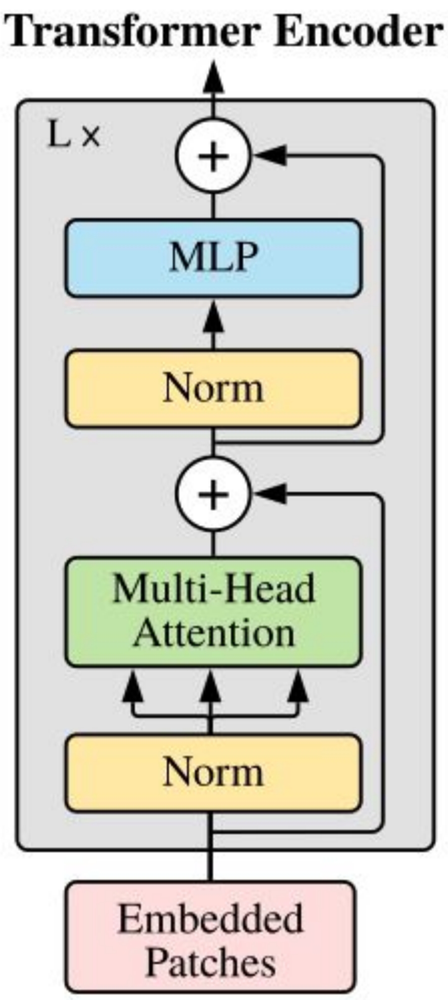
We use a 50 layer pre-trained ResNet model.

We finetune this model for our satellite imagery classification task (21 classes).

Learning Rate	**Weight Decay**	**Epochs**	**Loss function**	**Optimizer**
3e-4	1e-4	100	Cross Entropy Loss	SGD

Ref:https://medium.com/data-science/the-annotated-resnet-50-a6c536034758

Vision Transformer - DeiT



We use a 12 layer pre-trained DeiT model (distilled version of ViT-Base).

We finetune this model for our satellite imagery classification task (21 classes).

Learning Rate	Weight Decay	Epochs	Loss function	Optimizer
3e-4	1e-4	50	Cross Entropy Loss	SGD

Ref: https://arxiv.org/pdf/2010.11929

Evaluation Metrics

- *Evaluation Metrics*

Accuracy	**Precision**	**Recall**	**F1-score**

Since our dataset is **fully balanced**, hence **accuracy** is considered to be the optimal choice to evaluate our model. We also plot **confusion matrix** on the test set. Hence, all of these provide a thorough metrics evaluation in our project.

Results - Baseline CNN

Analysis Across Multiple Data Transformations

Data Transformation	**Accuracy**
Random Horizontal Flip	**74%**
Random Rotation	**74%**
Random Resized Crop	68%
Color Jitter	65%
Random Affine	72%
Random Perspective	69%

We choose **Random Horizontal Flip and Rotation** as final data transformation techniques.

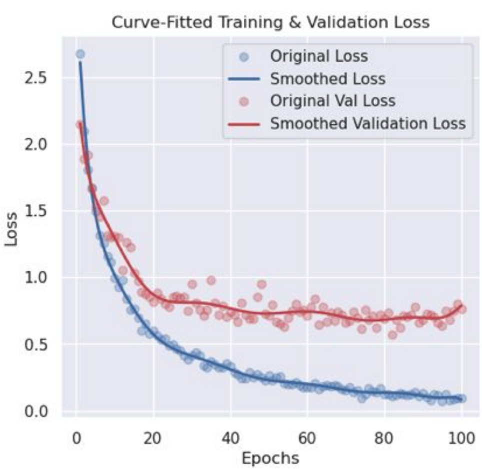
Adam V/S SGD Optimizer Analysis

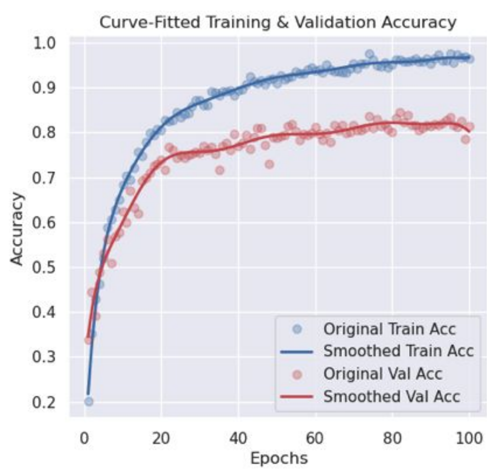
Optimizer	Accuracy	Precision	Recall	F1 Score
SGD	**83.05%**	**83.14%**	**83.65%**	**82.66%**
Adam	80.38%	80.93%	80.99%	80.21%

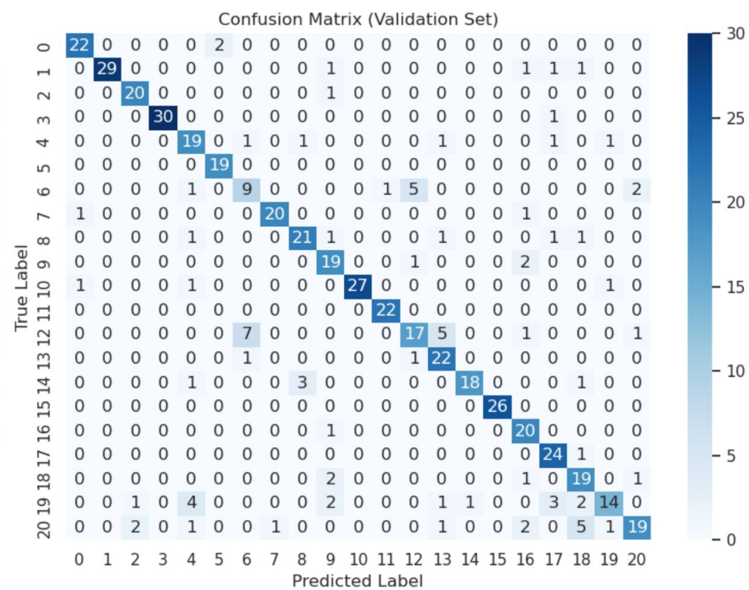
Using the best 2 data transformations, we analyse which optimizer is best - **SGD performs the best for our task!**

Loss-Accuracy Curves and Confusion Matrix

Using the best hyperparameters we plot the loss-accuracy curves and the confusion matrix (SGD + 2 data transformations + other parameters as discussed previously)







Test-Set Performance:

Accuracy - 83.05%, Precision - 83.14%, Recall - 83.65%, F1-score - 82.66%

Note: Since it is overfitting at later epochs we use the validation set to find best model, and then use the best model for test-set

Performance Time & Trainable Parameters Count

- Total Trainable Parameters: 920,469
- Per Epoch Train Time - 1.6 secs

All the models are also trained on H100 GPU.

Results - ResNet50

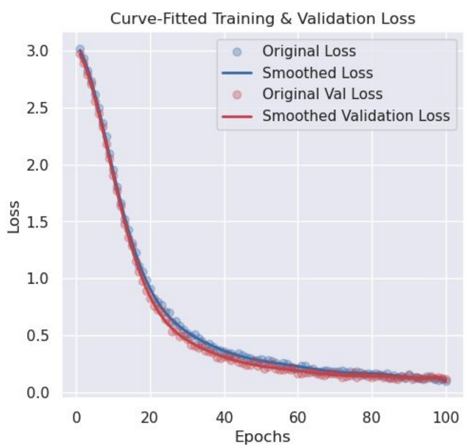
Unfreezing Layers in Group & Trainable Parameters, Train Time per Epoch

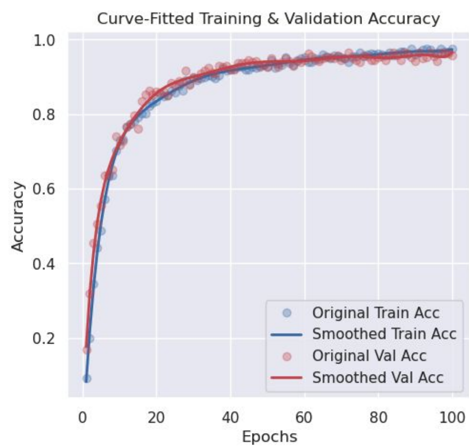
Unfreezed (Trainable) Components	Test Accuracy	Total Trainable Parameters	Train Time Per Epoch
All 4 stages + CH	**96.51%**	**23,551,061**	**2.87s**
Last 3 stages + CH	93.02%	23,335,253	2.79s
Last 2 stages + CH	94.29%	22,115,669	2.71s
Last 1 stage + CH	91.43%	15,017,301	7.64s
Only CH	83.17%	52,565	6.83s

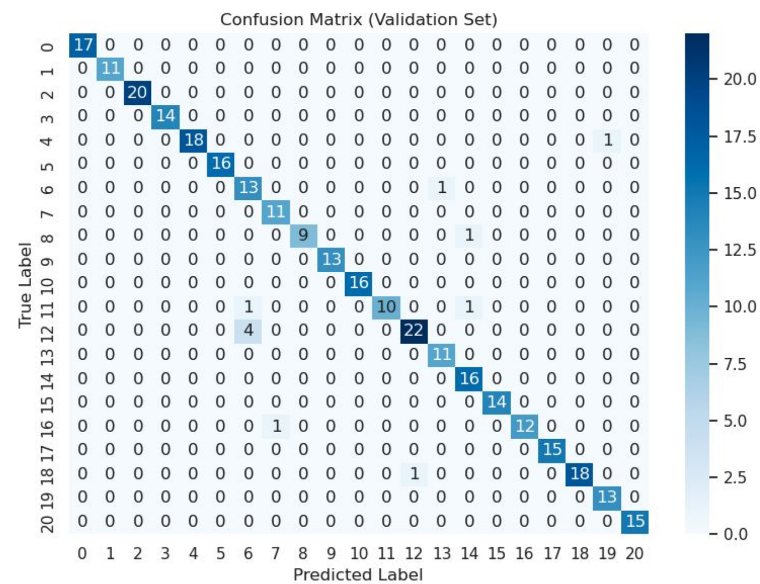
- **Unfreezing all layers** give the best performance but also has higher train time and resource utilization. Also, the performance gain is very good (13% up) in comparison to the baseline CNN accuracy.

Loss-Accuracy Curves and Confusion Matrix

Using the best hyperparameters we plot the loss-accuracy curves and the confusion matrix (SGD + 2 data transformations + other parameters as discussed previously) for the completely unfrozen model







Test-Set Performance:

Accuracy - 96.51%, Precision - 96.81%, Recall - 96.79%, F1-score - 96.62%

Note: Since it is overfitting at later epochs we use the validation set to find best model, and then use the best model for test-set

Results - DeiT

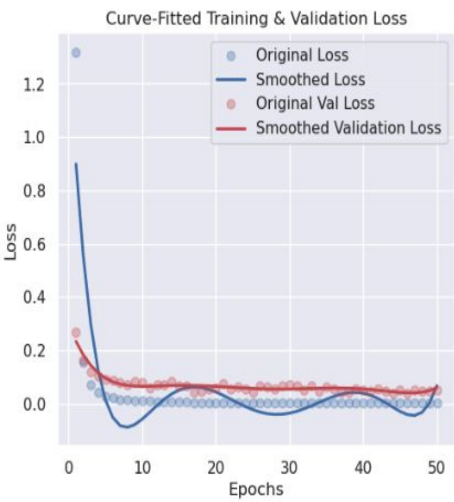
Unfreezing Layers in Group & Trainable Parameters, Train Time per Epoch

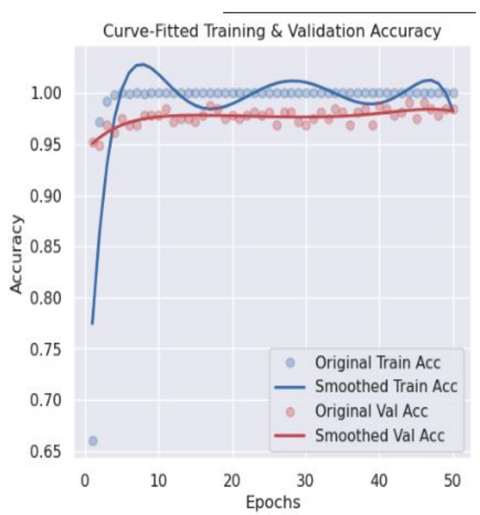
Unfreezed (Trainable) Components	Test Accuracy	Total Trainable Parameters	Train Time Per Epoch
All 12 layers + Classification Head (CH)	**98.1%**	**85,816,341**	**9.7s**
Last 9 layers + CH	97.14%	64,552,725	9.02s
Last 6 layers + CH	96.83%	43,289,109	8.36s
Last 3 layers + CH	95.87%	22,025,493	7.64s
Only CH	95.56%	761,877	6.83s

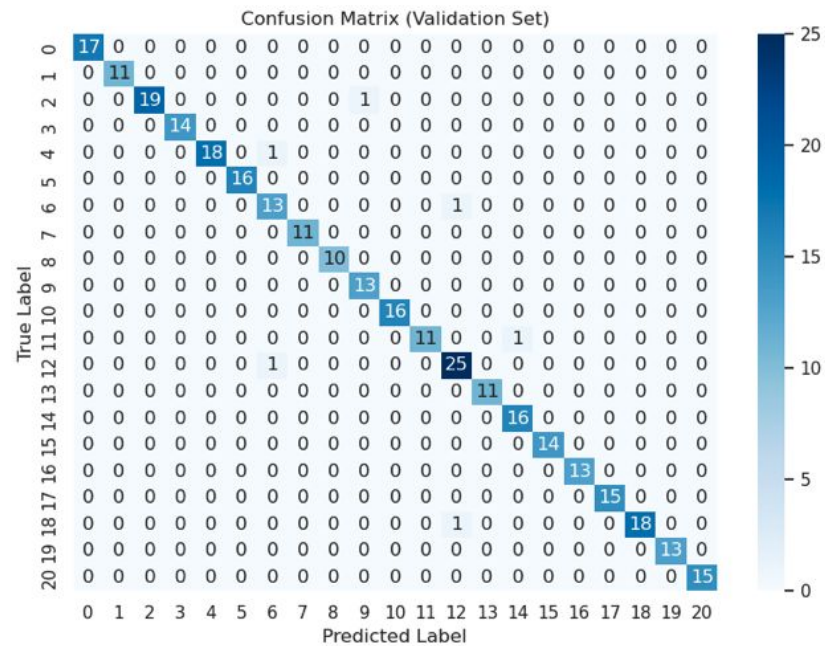
- **Unfreezing all layers** give the best performance but also has higher train time and resource utilization.
- Furthermore, we observe that just by unfreezing classification head (CH) we can achieve almost 96% test-accuracy.

Loss-Accuracy Curves and Confusion Matrix

Using the best hyperparameters we plot the loss-accuracy curves and the confusion matrix (SGD + 2 data transformations + other parameters as discussed previously) for the completely unfrozen model







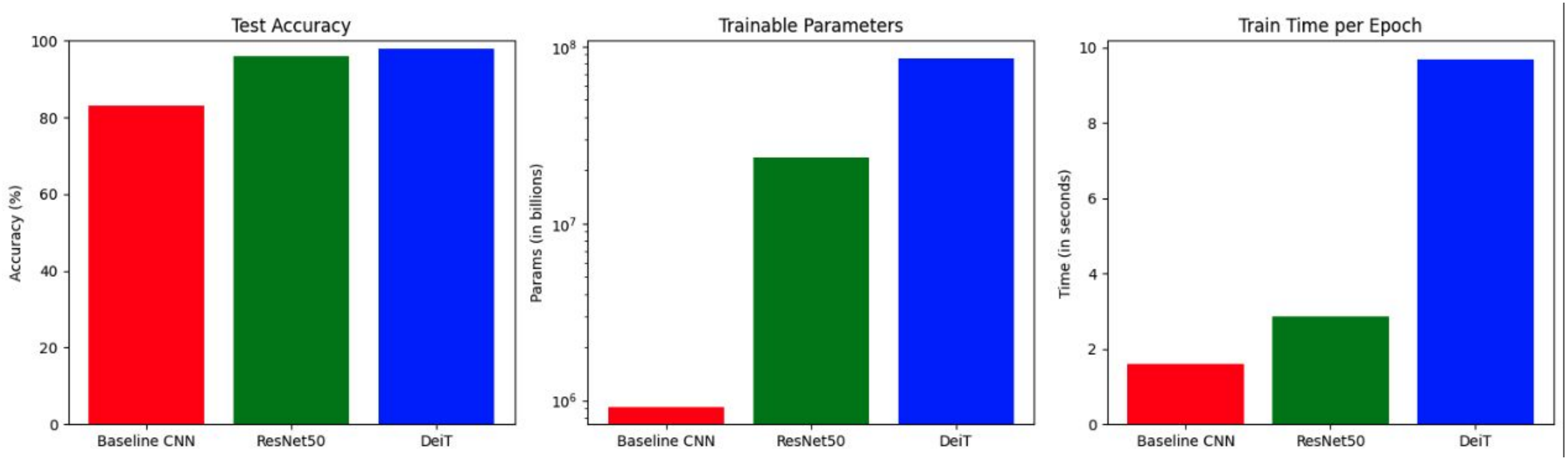
Test-Set Performance:

Accuracy - 98.1%, Precision - 98.39%, Recall - 98.34%, F1-score - 98.33%

Note: Since it is overfitting at later epochs we use the validation set to find best model, and then use the best model for test-set

Final Analysis

Test Set Performance and Efficiency Comparison across the 3 Models



- ResNet50 and DeiT outperforms the Baseline model significantly (13-15%)
- ResNet50 and DeiT have almost similar performance
- Number of trainable parameters for ResNet50 and train-time per epoch much lesser than that of DeiT.
- **ResNet50 seems to be the ideal model choice.**

References

[1] UC Merced Land Use Dataset. Available: https://www.kaggle.com/datasets/abdulhasibuddin/uc-merced-land-use-dataset/data.