Foundations of Deep Learning Project

Team ID:LTVIP2025TMID38414
Ayeluri Agastya subhramayam sri Harsha

Steps

- Understanding of the Dataset
- Loading and Sampling
- EDA: Exploratory Data Analysis
- Preprocessing + Splitting data into Training and Validation Set
- Classification: Neural Network from scratch and Transfer Learning
- Selection and Comparison of the Best Models
- Fitting on Test data and Best Model Choice

Rice Image Dataset

- https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset
- 75K images.
- 5 types of rice grains:

Arborio

Basmati

Ipsala

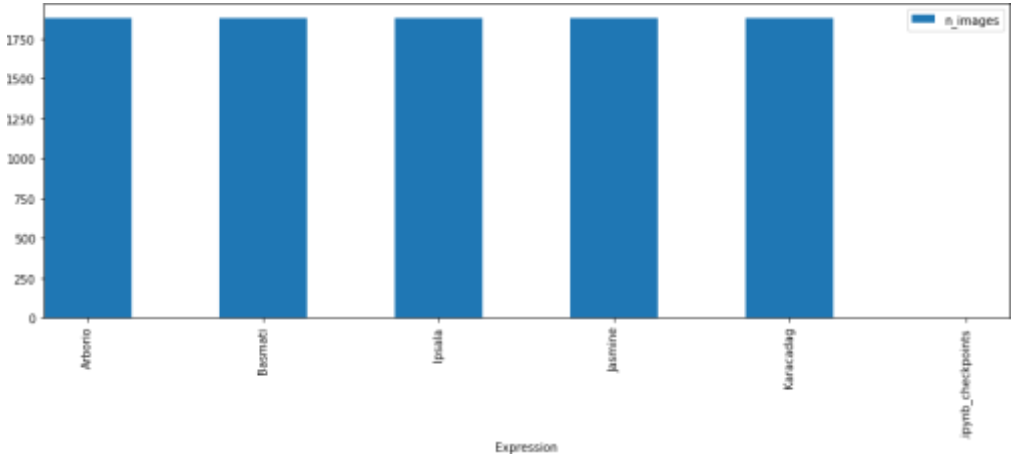
Jasmine

Karacadag

- 15K images for each type of rice grain
- Task: Rice grain classification

Sampling

Target classes are equidistributed.



75K images → 15K images for each type of rice

Sampling to train NN from scratch and pre-trained NN with transfer learning

5K images → 1K images for each type of rice

Sampling to train best NN from scratch and best pre-trained NN with transfer learning

10K images → 2K images for each type of rice

Neural Network from scratch

- **Step to define a NN from scratch**
- **Pre-processing** + Splitting data into training and validation: 80% of the data is contained in training set, 20% of the data is contained in the validation set

  Pre-processing for training set: standardization + data augmentation

  Pre-processing for validation set: standardization
- **NN Architecture definition**
- **NN Compilation**: definition of network macro-elements (optimizer and learning rate, loss function, additional metrics, parameters, etc.)
- **NN Training**
- **Summary and Results visualization**

Model name Net name	Network Architecture + Properties	Accuracy	Notes
Modelv1 net	2 CNN (32, 64)	*Acc* 0.95 *Val_Acc* 0.88	From the 30th epoch onward the model remains stable. Despite having very good performance and there seems to be no presence of overfitting it is decided to increase the stability of the model in the next steps.
Modelv2 net2	2 CNN (32, 64) + batch normalization	*Acc* 0.97 *Val_Acc* 0.88	Very unstable model. The complexity of the convolutional layer is increased.
Modelv3 net3	3 CNN (32, 64, 128) + batch normalization	*Acc* 0.98 Val_Acc: 0.86	Very unstable model. Accuracy decreased. The regularization term is added.
Modelv4l1 net4l1	3 CNN (32, 64, 128) + batch normalization + l1 (lasso regularization)	*Acc* 0.97 *Val_Acc* 0.65	Very unstable model. As in the previous cases the Accuracy (on training) is very high while on validation it is lower.
Modelv4l2 net4l2	3 CNN (32, 64, 128) + batch normalization + l2 (ridge regularization)	*Acc* 0.98 *Val_Acc* 0.88	Accuracy on validation is higher than that of the previous model (lasso regularization). The model is more stable in the last 5 epochs (35th onwards). Complexity in the fully connected part is increased.
Model4l2_5 net4l2_5	3 CNN (32, 64, 128) + batch normalization + l2 (ridge regularization) + early stopping	*Acc* 0.97 *Val_Acc* 0.92	Accuracy increased.
Model4_5 net4_5	3 CNN (32, 64, 128) + batch normalization + l2 (ridge regularization) + 1 Dense (64)	*Acc* 0.98 *Val_Acc* 0.81	Very unstable model. Accuracy on validation has decreased significantly compared to that obtained in Modelv4l2. However, it is tried to increase the complexity of the fully connected layer again.
Modelv5 net5	3 CNN (32, 64, 128) + batch normalization + l2 (ridge regularization) + 2 Dense (128, 64)	*Acc* 0.98 *Val_Acc* 0.82	Accuracy has improved by 1%. Despite this improvement the performance of Modelv4l2 (with only output dense layer) seems to be better. We change the weight initialization algorithm in the convolutional layers. We insert the He initializer because it might work better in convolutional layers where the activation function is a ReLU.
Modelv6 net6	3 CNN (32, 64, 128) + batch normalization + l2 (ridge) + He weights inizialization	*Acc* 0.98 *Val_Acc* 0.93	The algorithm converges very early. The final performance at the 40th epoch is very high, compared with previous networks. Nevertheless, it turns out to be somewhat unstable in the later epochs. Early Stopping mechanism is implemented.
Modelv6_5 net6_5	3 CNN (32, 64, 128) + batch normalization + l2 (ridge) + He weights inizialization + early stopping	*Acc* 0.98 *Val_Acc* 0.95	The reducing learning rate mechanism is added to try to make the model more stable and increase its performance.
Modelv7 **net7**	3 CNN (32, 64, 128) + batch normalization + l2 (ridge) + He inizialization + Reduce learning rate	*Acc* 0.996 *Val_Acc* 0.98	The model achieves very good performance on both training and validation. In addition, the model is very stable in the 20th epoch onwards.

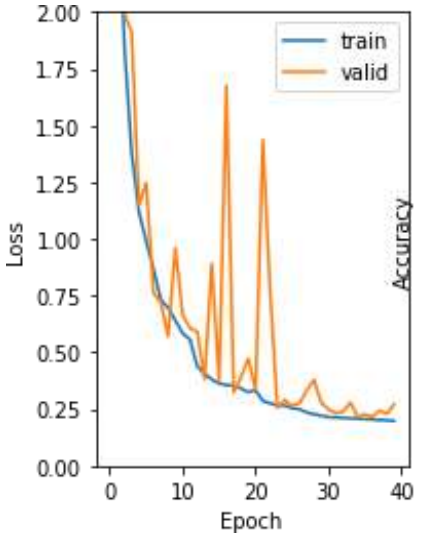
Best Model

Architecture

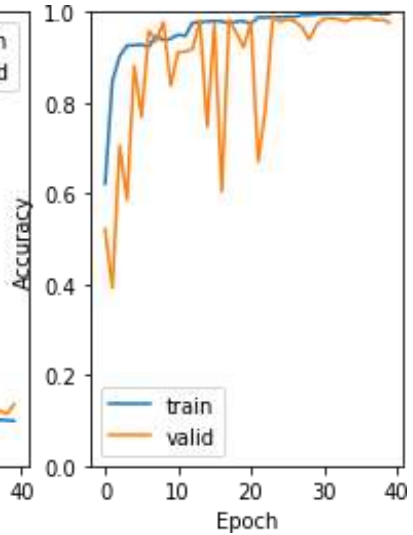
- Input Layer
- 2D convolutional layer: 3x3 filter, 32 neurons, Ridge regularizer, He uniform inizializer
- ReLU activation function
- Batch Normalization layer
- Max Pooling layer (3x3)
- 2D convolutional layer: 3x3 filter, 64 neurons, Ridge regularizer, He uniform inizializer
- ReLU activation function
- Batch Normalization layer
- Max Pooling layer (3x3)
- 2D convolutional layer: 3x3 filter, 128 neurons, Ridge regularizer, He uniform inizializer
- ReLU activation
- Batch Normalization layer
- Global Max Pooling layer
- Output layer: Softmax activation function, Ridge regularizer

Callback: Reduce Learning Rate

Training and Validation Loss



Training and Validation Accuracy



Transfer Learning

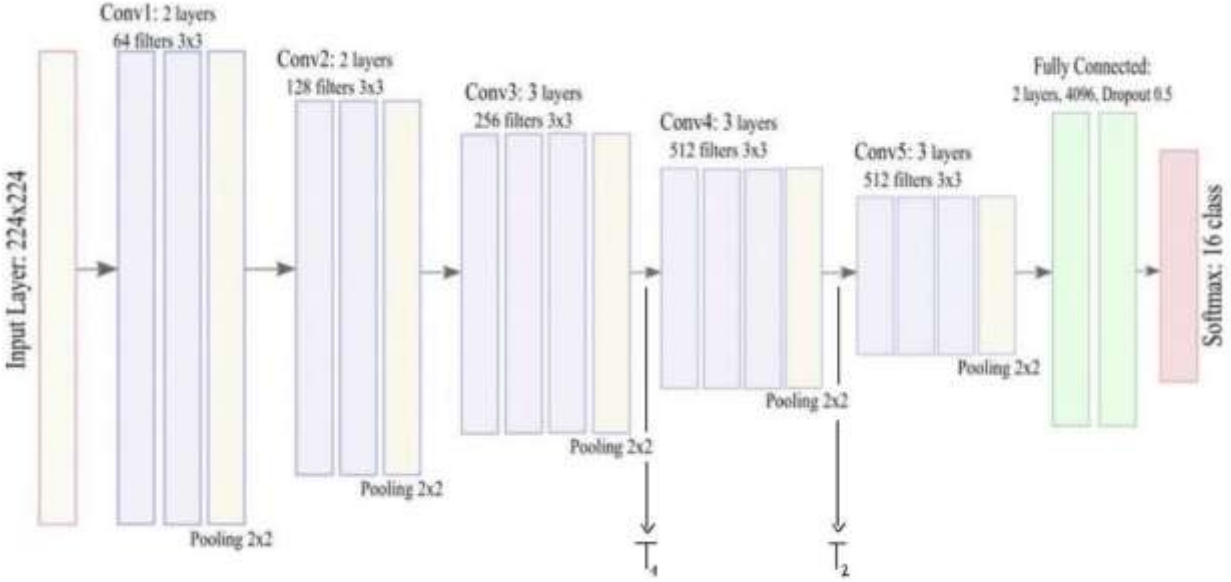
Steps:

- Preprocess Data: The images are converted from RGB to BGR, then each color channel is zero-centered with respect to the ImageNet dataset, without scaling.
- Loading of the pre-trained network.
- Freeze the convolutional base before compiling and train the model.
- Define a FNN architecture.
- Callbacks.
- Compile and train.

Applications Results

Model name Net name	Network Architecture + Properties	Accuracy	Notes
ResNet50	Pre-trained Network + FFN + Dropout + CallBacks (EarlyStopping, ReduceLrOnPlateau)	*Acc*: 0.97 *Val_Acc*: 0.98	The network performs very well and it converges at the 14th epoch without ever overfitting.
MobileNet	Pre-trained Network + FFN + Dropout + CallBacks (EarlyStopping, ReduceLrOnPlateau)	*Acc*: 0.88 *Val_Acc*: 0.85	The performance is slightly worse than the previous network. In addition, the network tends to overfit.
Vgg16	Pre-trained Network + FFN+Droput + CallBacks (EarlyStopping, ReduceLrOnPlateau)	*Acc*: 0.83 Val_Acc: 0.85	Although the performance is the worst among trained networks, the trend of loss and accuracy curves is excellent.
Vgg16 cut1	Pre-trained Network + Avg global pooling + FFN + Dropout + CallBacks (EarlyStopping, ReduceLrOnPlateau)	*Acc:* 0.87 *Val_Acc:* 0.95	Accuracy values are improved with a cut to block3_pool.
Vgg16 cut2	Pre-trained Network + Avg global pooling + FFN + Dropout + CallBacks (EarlyStopping,)	*Acc:* 0.96 *Val_Acc:* 0.97	Accuracy values are definitely improved with a cut to block4_pool.

VGG16 T1|T2


Input Layer: 224x224
Conv1: 2 layers
64 filters 3x3
Pooling 2x2
Conv2: 2 layers
128 filters 3x3
Pooling 2x2
Conv3: 3 layers
256 filters 3x3
Pooling 2x2
Conv4: 3 layers
512 filters 3x3
Pooling 2x2
Conv5: 3 layers
512 filters 3x3
Pooling 2x2
Fully Connected:
2 layers, 4096, Dropout 0.5
Softmax: 16 class
T1
T2

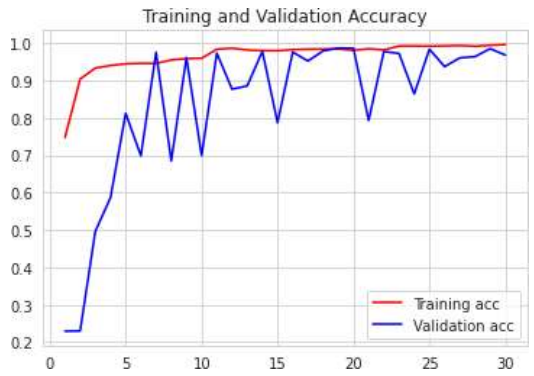
Best model

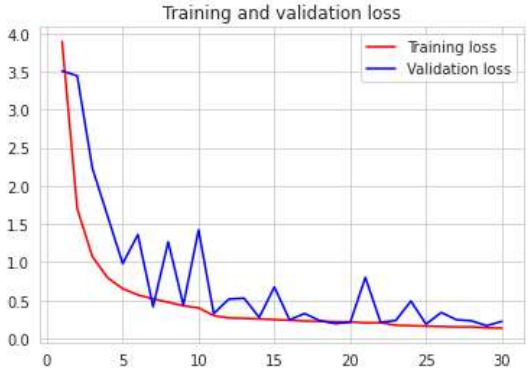
Architecture:

- ResNet50.
- Dense layer 2048 neurons, activation function "relu".
- Dense layer 512 neurons, activation function "relu".
- Dropout layer 0.15.
- Dense layer 128 neurons, activation function "relu".
- Dense layer 64 neurons, activation function "relu".
- Dropout layer 0.15.
- Dense layer 32 neurons, activation function "relu".
- Dense layer 16 neurons, activation function "relu".
- Output layer 5 output neurons, activation function "softmax".

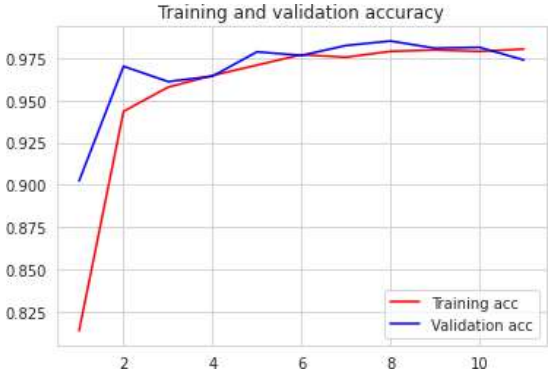
Best Models Evaluation on 10k dataset

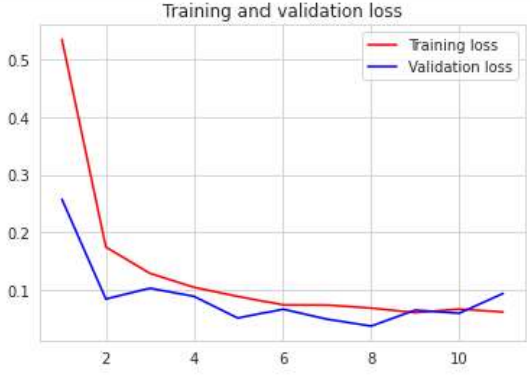
Best Model from Scratch





Best Model with Transfer Learning





Classification Report on Test set

Best Model from scratch

	Precision	Recall	F1 score	Support
Arborio	1.000	0.860	0.925	100.00
Basmati	0.980	1.000	0.990	100.00
Ipsala	0.943	1.000	0.971	100.00
Jasmine	0.916	0.980	0.947	100.00
Karacadag	1.000	0.990	0.995	100.00
Accuracy	0.966	0.966	0.966	0.966
Macro Avg	0.986	0.966	0.966	500.00
Weighted Avg	0.986	0.966	0.966	500.00

Best Model with Transfer Learning

	Precision	Recall	F1 score	Support
Arborio	0.971	1.000	0.985	100.00
Basmati	0.962	1.000	0.980	100.00
Ipsala	1.000	1.000	1.000	100.00
Jasmine	1.000	0.950	0.974	100.00
Karacadag	1.000	0.980	0.990	100.00
Accuracy	0.986	0.986	0.986	0.986
Macro Avg	0.986	0.986	0.986	500.00
Weighted Avg	0.986	0.986	0.986	500.00

Thanks for your attention