NYCU Pattern Recognition, Homework 4

[312551814], [DAHMANI Malik]

Part. 1, Kaggle (70% [50% comes from the competition]):

(10%) Implementation Details

My implementation integrates image processing, feature extraction using a deep learning model, and classification using traditional machine learning algorithms. The featurs I implemented:

- Data loading: Images are loaded from pickle files stored in class-specific directories using the `load_images_from_pkl` function. The `load_data` function consolidates images and their respective labels into arrays.

- Feature Extraction: I used EfficientNetB0 model (pre-trained on ImageNet) to extract features from the images. The top layer of the model is removed, and a global average pooling layer is added. The `extract_features` function preprocesses images, generates features using the EfficientNet model, and flattens the output.

- MIL Attention Pooling: An attention mechanism is implemented in the `mil_attention_pooling` function to potentially enhance feature selection.

- Training and Evaluation: Extracted features are split into training and validation sets using `train_test_split`. Three classifiers (RandomForest, SVC, MLP) are trained on the extracted features. The best classifier is selected based on validation accuracy.

- Prediction and Submission: Features from the test set are extracted using the same EfficientNet model. The best classifier is used to predict the test set labels. A submission file is created with predictions.

For the hyperparemeters, I used:

- EfficientNetB0 Model

- Attention Mechanism: A dense layer with softmax activation is applied to input features to generate attention weights. Element-wise multiplication of input features with attention weights to create attended features.

-Classifiers: RandomForestClassifier(n_estimators=100, random_state=42), SVC(kernel='linear', random_state=42) and MLPClassifier(hidden_layer_sizes=(100,), max_iter=1000, random_state=42).

My training strategy consist of:
- Data Preparation: Images are preprocessed using EfficientNet's `preprocess_input` function. Extracted features are flattened for compatibility with traditional classifiers.

- Splitting Data: The dataset is split into training and validation sets in an 80-20 ratio.

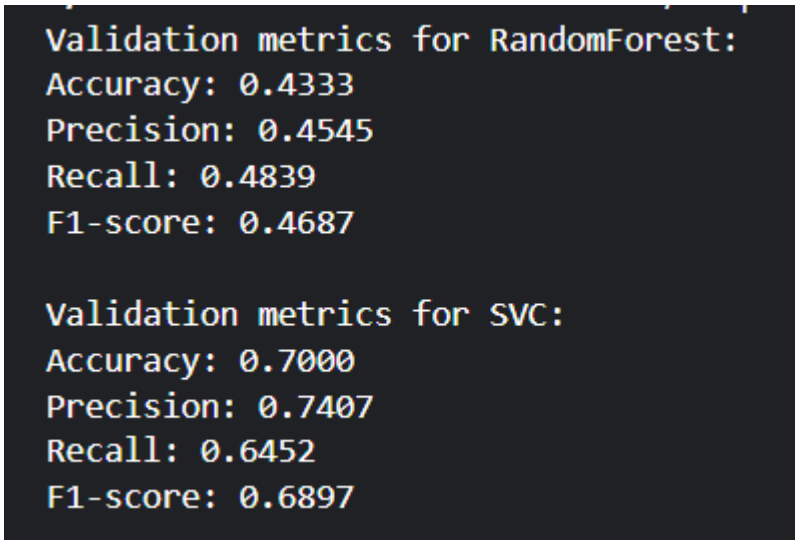
- Model Training: Each classifier is trained on the training set. Validation accuracy is calculated for each classifier to evaluate performance.

- Model Selection: The classifier with the highest validation accuracy is selected as the best model.

- Saving and Predicting: The best classifier is saved using `pickle`. Predictions on the test set are made using the best classifier. A submission file is generated in CSV format for evaluation or competition purposes.

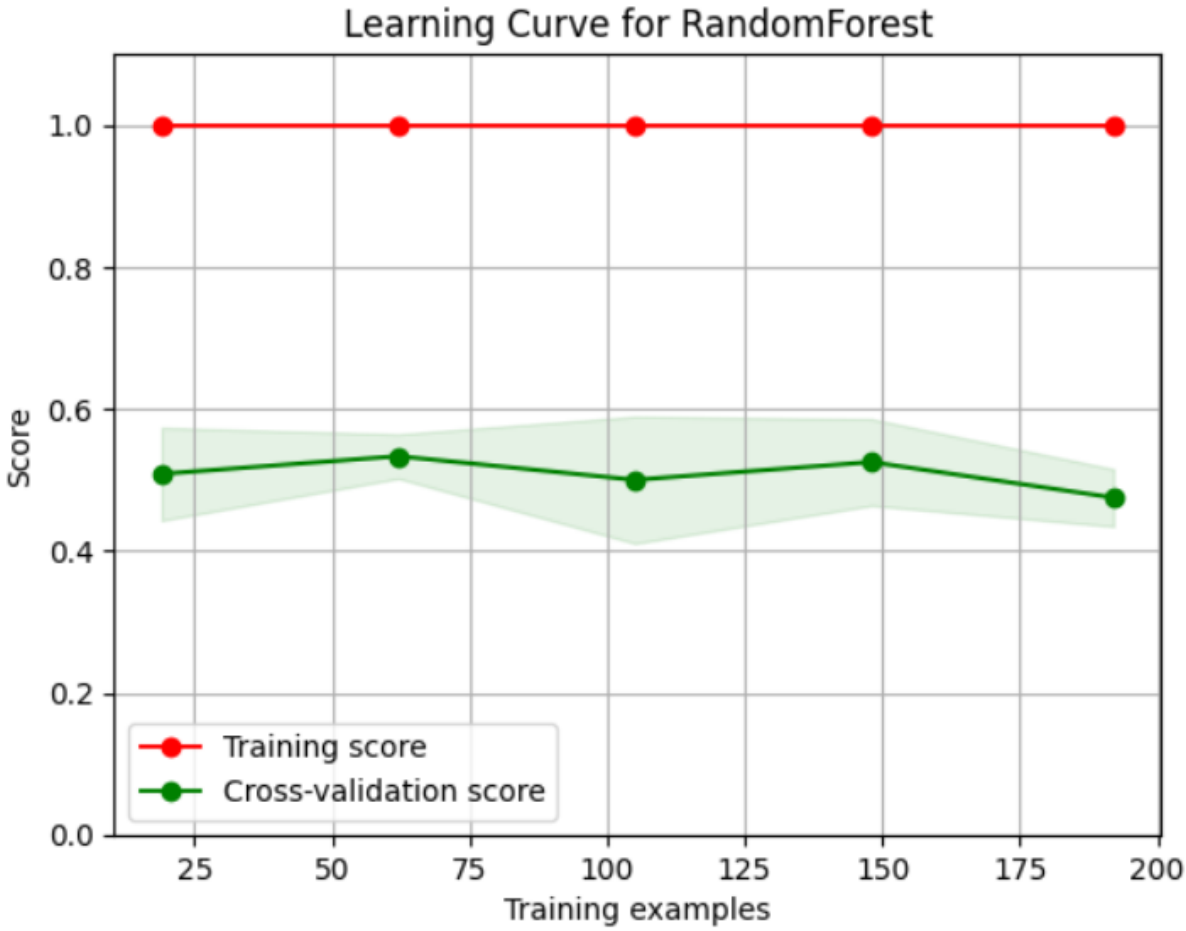
(10%) Experimental Results

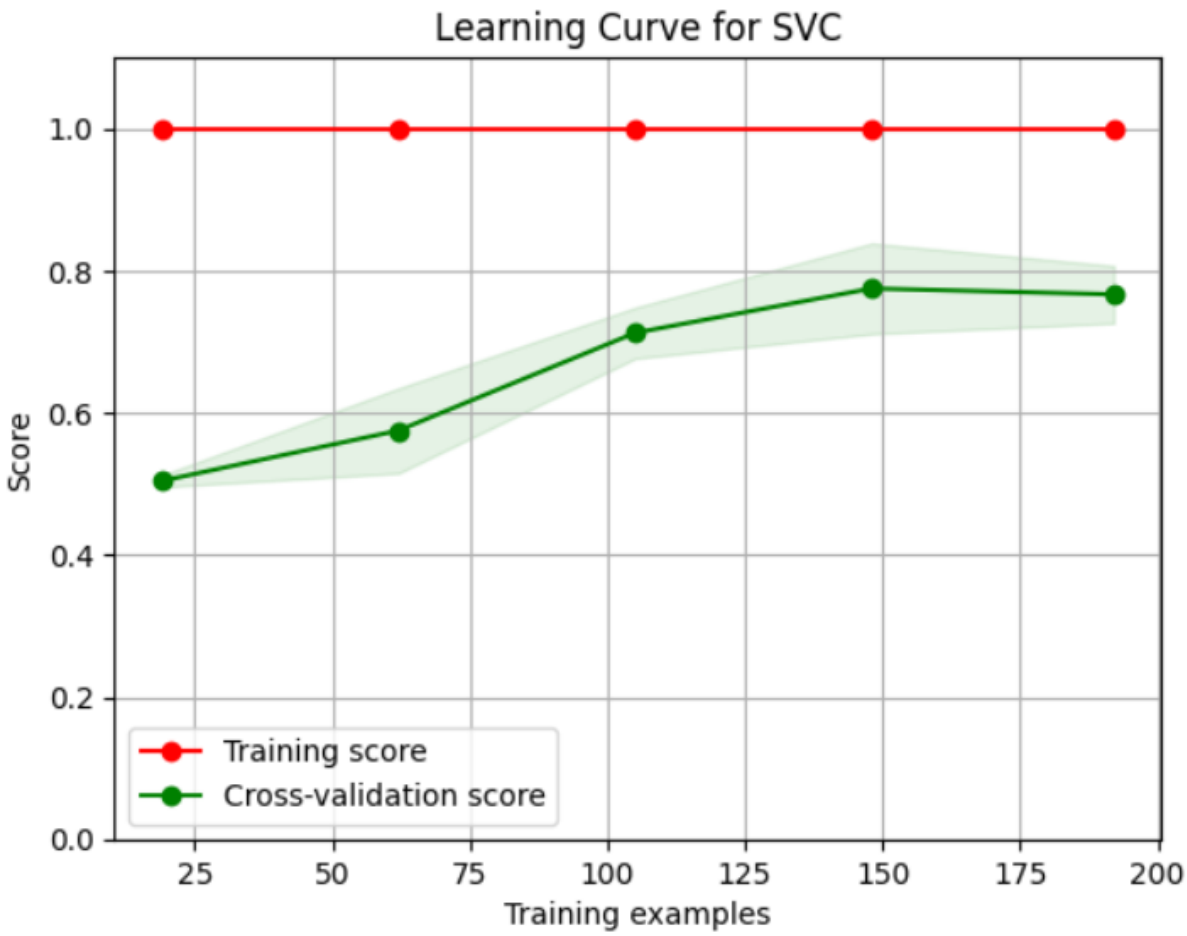
The validation metrics for my model:

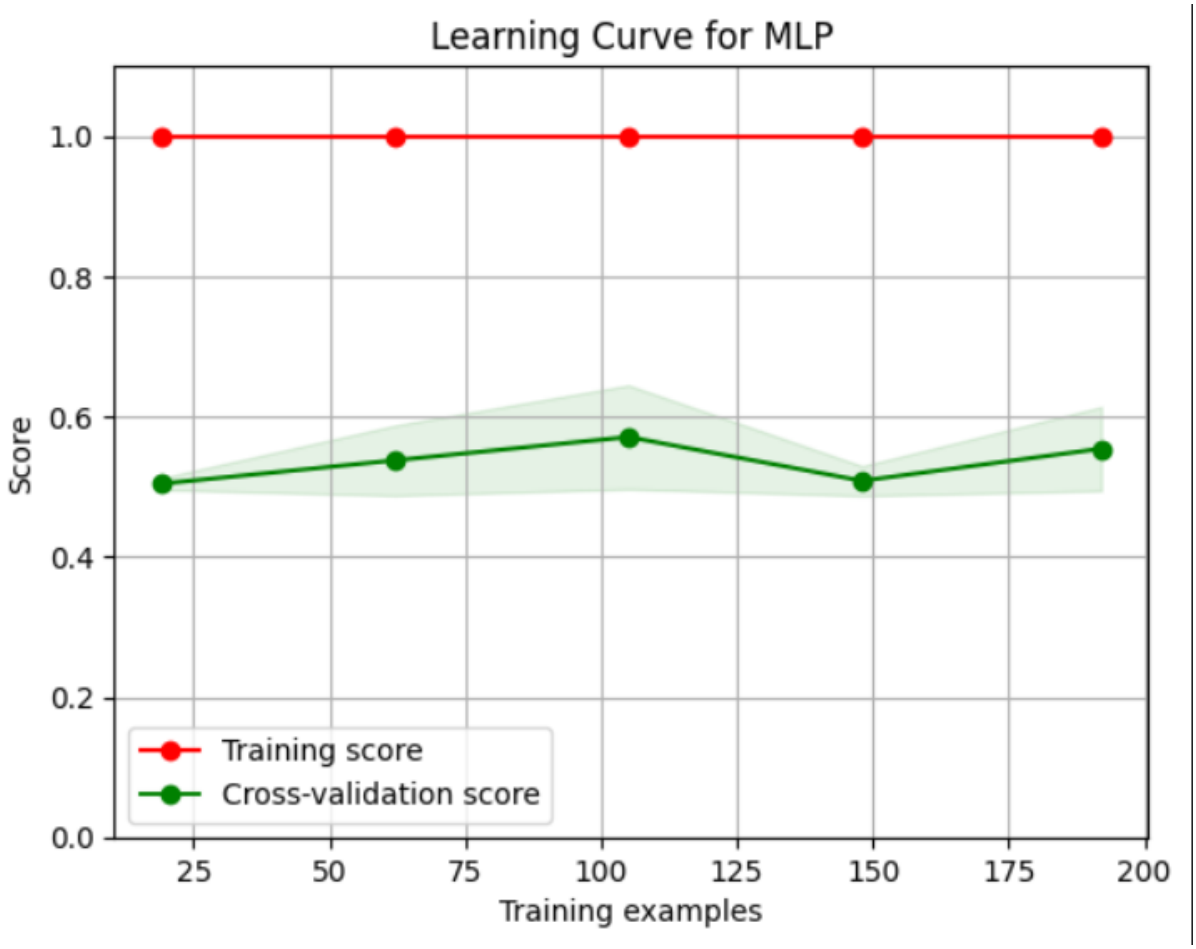


```
Validation metrics for MLP:
Accuracy: 0.4833
Precision: 0.0000
Recall: 0.0000
F1-score: 0.0000
```

The learning curves for my model:




Learning Curve for SVC
Score
Training examples
Training score
Cross-validation score
1.0
0.8
0.6
0.4
0.2
0.0
25
50
75
100
125
150
175
200



For the ablation study, I did an ablation study by removing the last featur e. The accuracy on the training set was

```
Ablation study accuracy with feature modification: 0.7000
```

Part. 2, Questions (30%):

1. (10%) Why Sigmoid or Tanh is not preferred to be used as the activati on function in the hidden layer of the neural network? Please answer in detail.

Sigmoid and Tanh are popular activation functions but they’ re not preferred in th e hidden layer of the neural network for various reasons. The main reason is that Sigmoid and Tanh function saturate when their inputs are very large or very small leading to gradie nts close to zero. When the gradients are closed to zeros it’ s called Vanishing gradient.

The vanishing gradient problem is a recurring problem when implementing gradien t descent. This problem occurs when the gradient becomes very small. As the gradient bec omes very small, the change in weights becomes minimal, preventing the training from co

nverging to the optimal solution. As a result, learning becomes slower and slower. In a neural network, this means that the first layers of the network learn more slowly than the last.

Moreover, sigmoid function output positive values only. The output range of (0, 1) is not centered around zero, which can make the optimization process more difficult.

To continue, the tanh function is computationally more expensive compared to other functions like ReLU.

2. (10%) What is overfitting? Please provide at least three techniques with explanations to overcome this issue.

Overfitting can be defined as the fact that a machine learning model perform well on the training data but fails to generalize to unseen data (test data). The model learns to capture particular noise in the training data and not the pattern. Thus, the model doesn't perform well on new data.

To overcome overfitting, you can use:

- Cross-Validation: Cross-validation is a technique used to assess the performance of a machine learning model. Instead of training the model on the entire dataset, the data is divided into multiple subsets or folds. The model is trained on a subset of the data and evaluated on the remaining subsets. This process is repeated multiple times, with each subset used as both training and validation data. By averaging the performance across multiple folds, cross-validation provides a more reliable estimate of the model's performance on unseen data. It helps to detect overfitting by evaluating the model's performance on different subsets of the data.

- Regularization: Regularization techniques are used to prevent overfitting by adding a penalty term to the loss function. This penalty term discourages complex models that are more likely to overfit. By adding these penalty terms, regularization encourages simpler models that generalize better to unseen data.

- Feature Selection: Feature selection involves selecting a subset of the most relevant features from the dataset while discarding irrelevant or redundant features. This helps to reduce the complexity of the model and prevent overfitting.

3. (15%) Given a valid kernel $k_1(x,x')$, prove that the following proposed functions are or are not valid kernels. If one is not a valid kernel, give an example of $k(x,x')$ that the corresponding K is not positive semidefinite and show its eigenvalues.

a. $k(x, x') = k_1(x, x') + \|x\|^2$

b. $k(x, x') = k_1(x, x') - 1$

c. $k(x, x') = k_1(x, x') + \exp(x^T x')$

d. $k(x, x') = \exp\big(k_1(x, x')\big) - 1$

a/ For $k(x, x') = k_1(x, x') + \|x\|^2$, we know that $k_1(x, x')$ is positive semi-definite. Moreover, $\|x\|^2$ is a non-negative scalar. Considering $\lambda_1, \lambda_2, ..., \lambda_n$ the eigenvalues of k_1, the eigenvalues of K will be : $\lambda_1 + \|x\|^2, \lambda_2 + \|x\|^2, ..., \lambda_n + \|x\|^2$.

$\|x\|^2$ being non-negative, the eigenvalues of k will be at least as large as the eigenvalues of k_1. Thus k will be positive semidefinite and $k(x, x') = k_1(x, x') + \|x\|^2$ a valid kernel function.

b/For $k(x, x') = k_2(x, x') - 1$, we supose that $k_2(x, x') = (x^t x')^2$ (wich is a valid kernel seen in class) and $x = [1, 0]^t$ and $y = [0, 1]^t$:

$k_2(x, x') = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

This matrix is positive definite because it's a diagonal matrix with positive values on the diagonal.

Now for k we have :

$k(x, x') = k_2(x, x') - 1 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$

The characteristic equation is :

$\det(k(x, x') - \lambda I) = \det(\begin{bmatrix} -\lambda & -1 \\ -1 & -\lambda \end{bmatrix}) = \lambda^2 - 1$

By solving this equation we have $\lambda_1 = 1 and \lambda_2 = -1$. Thus k is not positive semidefinite. We can then conclude that $k(x, x') = k_2(x, x') - 1$ is not a valid kernel

c/ For $k(x, x') = k_3(x, x') + e^{x^t x'}$, given that k_3 is a valid kernel, we know that it is positive semidefinite. Now we need to prov that $e^{x^t x'}$ retains this. We know that :
$e^{x^t x'} = \sum_{n=0}^{\infty} \frac{(x^t x')^n}{n!}$

Each term of this sum is a polynomial kernel nd is positive semidefinite. Moreover, a finite sum of positive semidefinite matrices is positive semidefinite, and this property extends to an infinite sum if the series converges. Thus $e^{x^t x'}$ is positive semidefinite and a valid kernel.

By property (6.17) in the kernel chapter, $k(x, x') = k_3(x, x') + e^{x^t x'} = k_3(x, x') + k_4(x, x')$ is a valid kernel. We can then conclude that $k(x, x') = k_3(x, x') + e^{x^t x'}$ is a valid kernel.

d/ For $k(x, x') = e^{k_1(x,x')} - 1$, we know that $k_1(x, x')$ is positive semidefinite and so e^{k_1} is also positive semidefinite. Subtracting 1 from each entry of a positive semidefinite matrix corresponds to shifting its eigenvalues. The minimum of the eigenvalues would be $e^0 = 1$. Thus, since the smallest value of $e^{k_1(x,x')}$ is 1 when $k_1(x, x') = 0$, subtracting 1 still results in non-negative eigenvalues, preserving the positive semidefinite property. So, $k(x, x') = e^{k_1(x,x')} - 1$ is a valid kernel.