**EE4146 Project Report**

6-Class Image Classification

Student Information~

Group 14-

| 55871122 | JEYAKUMAR Onasis Melchior |
| --- | --- |
| 55778474 | MUKHERJI Meenakshi |
| 55956246 | NANDI Mitsumi |
| 55907774 | PATRA Yuvraj |
| 55849809 | TIWARI Shivi |

**Abstract**

Image classification is a vital application of machine learning and can be used in security systems, self-driving cars, traffic control systems, medical diagnosis and more. This project gives an elementary idea of how machine learning is utilised in image classifiers. A 6-Class Image Classifier ( 6 classes being mountain, forests, streets, glaciers, sea, and buildings) by attempting different methods of feature extraction and classification algorithms.

**Introduction**

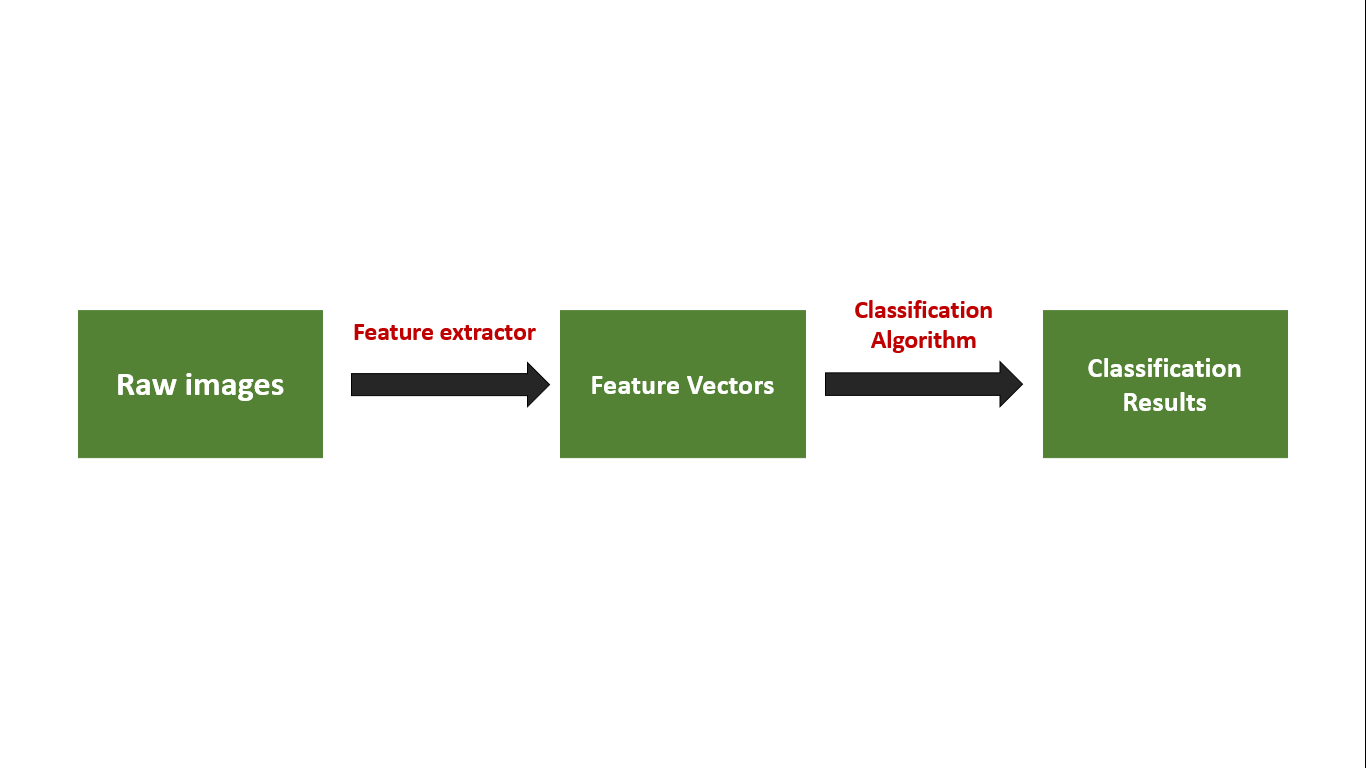
Classification of images is a critical issue in the computer vision field. Different deep learning algorithms solve a wide variety of problems related to classification. Their key feature is being able to process large and wide ranges of data sets, thus making them the prime technique for image classification. The aim of machine learning is to learn a classification function from a given set of features and training dataset. Deep learning algorithms perform the object feature extraction on their own automatically, and this engineering of the extraction of different types of object features is a heavy and time-consuming task. The deep learning algorithm provides the platform for changes in the deep learning model along with a wide variety of data sets. These tasks are, however, GPU heavy and sometimes even keep running for weeks. A popular opinion through tested and varied results is that image classification can be done easily with deep learning algorithms with minimal cost.

Our objective for this project is to perform a 6-class image classification, essentially find a suitable machine learning algorithm and apply it to solve the classification problem. The project also aims to optimise our algorithm and improve its accuracy by trying different classification algorithms. This project provides exposure to different image classification algorithms like CNN, Adaboosting, SVM, random forests, KNNs, multilayer perceptron, VGG, etc.

The project starts off with an aim to find better features and hence try to reduce the linear dimensionality using algorithms like PCA. PCA is known to be used for enhancing the accuracy of image classification and analysis. But this dimensionality reduction does not result in a very significant improvement in accuracy. Further, SVM linear classifier is tried since it has potential as a classifier for remotely sensed data, infact SVM is one of the best algorithms for image and pattern classification. The accuracy resulting from this model was considerably high. Moving on to the AdaBoosting classifier which is also popular for feature selection and classification, this algorithm maintains a set of weights over training data and adjusts them after each weak cycle of learning adaptively, its main idea is to maintain the collection of weights over the training data, it starts with all weights being equal and after every cycle, the incorrectly classified samples weight is increased, this is done to focus on the samples which are difficult to classify or train. With KNN, the accuracy level achieved was also not very promising although KNN algorithm is one the most simple algorithms for image classification since it essentially depends on the distance between the feature vectors. Random forest too did not give a competitive accuracy, random forest is essentially designed for the analysis of high dimensional data. After running the multilayer perceptron for this multi class image classification problem, it achieved the highest accuracy. This algorithm comprises three layers, the neurons inside the MLP are trained with the help of a back propagation learning algorithm, this algorithm is powerful for problems which are not separable linearly.

**Methods**

In this project, the aim was to classify images into 6-classes: buildings, mountains, street, sea, glacier and forest. To implement the model was trained on a given training dataset (with raw images) by extracting features and employing classification algorithms on the extracted features to classify the images (general pipeline shown below).



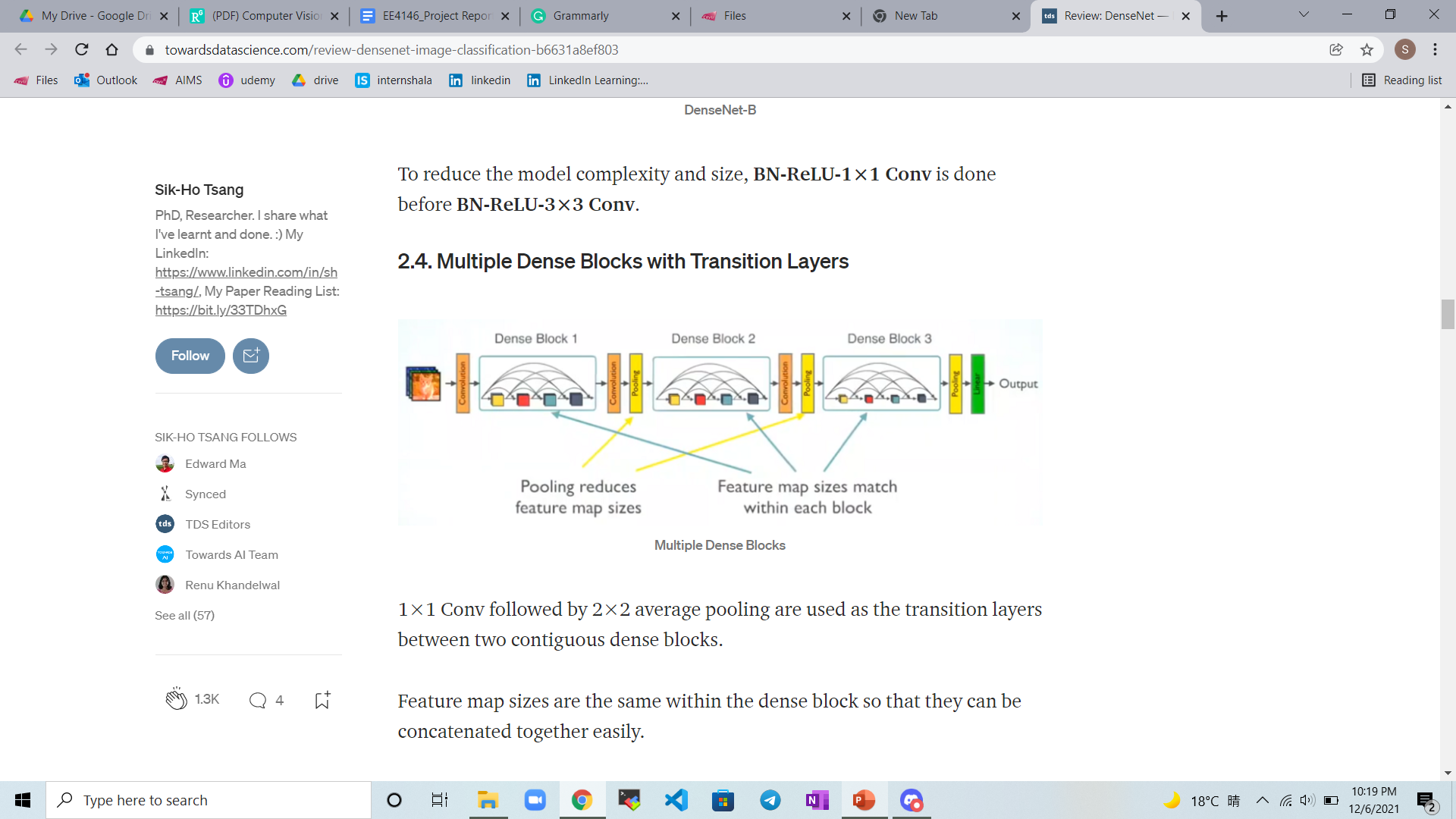
***Figure 1****: General Pipeline for 6-Class Image Classifier*

The models were evaluated using accuracy rate -

An emphasis was put on finding methods of feature extraction and classification which returned high levels of accuracy. This section elaborates on the feature extraction methods and classification algorithms used.

Some of the feature extraction methods used are-

1. DenseNet: The dense convolutional Network uses concatenation and multiple layers wherein the preceding layers passes its mapped features to the subsequent layers and hence additional inputs are received by each layer; it results in every layer receiving a collection of features from all the preceding layers. Concatenation results in the network layer being compact and this ultimately leads to a high efficiency in terms of computation and memory.



***Figure 2****: General architecture for DenseNet with multiple dense blocks*

In this project, DenseNet was used given its advantages of efficiency with regards to parameters and computation, maintenance of less complex features and inclusion of diverse features while executing its feature extraction.

1. ResNet50 : ResNet (Residual Networks), are built by stacking residual blocks and using “skipped connections”, wherein input of a layer is directly connected to the output after skipping a few connections. It solves the problem of degradation of accuracy as network depth increases. Increasing the number of layers generally leads to a decrease in training error. In our project, we used ResNet as it converges faster.

1. PCA : This method is very popular for dimensionality reduction. The reason why it was chosen for image classification is because while working with multiple images, a large amount of storage is required and PCA essentially helps in compressing and preserving the data.

For classification, the following methods were used -

1. Multilayer Perceptron: The multilayer perceptron provides a mapping (non linear) between input and corresponding output vector. It is one of the most widely used neural network structures. It was chosen for this project due to its high efficiency in classifying non linearly separable data. It incorporates a non linear activation function and performs static mapping.
2. Bagging classifier: Bagging comprises Bootstrapping and Aggregation. It assists in reduction of variance and allows multiple weak learners to get together against a single strong learner, essentially selecting the best prediction from a bunch of weak models. The advantage of avoidance of overfitting in bagging resulted in it being tested for the given problem.
3. Gradient Boosting: It is a type of ensemble method, which mainly consists of three components- loss function (to obtain an estimate of how accurate the model is), weak learner (classifiers with poor performance. e.g. decision trees), and an additive model (weak learners are added iteratively and sequentially to minimise loss function).

The advantages of this method includes its flexibility (as it can optimise many different loss functions and has various options for hyperparameter tuning). However it is computationally expensive and can have overfitting.

1. SGD: Stochastic Gradient descent is basically an optimization of Gradient descent to find the parameters of function to minimise the cost function. As opposed to calculating the derivative of the loss for all the data points, it is calculated for a single random data point, hence making it faster than gradient descent.
2. SVC: Support Vector Classifier implements discriminative classification by separating the classes using a line (for two-dimensional data) or a manifold (for multi-dimensional data) such that the margin is maximised. The hyperplanes are generated iteratively, and the best hyperplane is chosen. A kernel function is used to transform input data space into the form required - linear kernel for linear input space, polynomial kernel for distinguishing non-linear/curved input space, radial basis function (RBF) kernel for mapping to an indefinite dimensional space.
3. Random Forest: Random Forest is derived from the concept of decision trees, it essentially depends on several self learning decision trees and this leads to more robustness in results compared to one single decision tree.
4. Decision Tree: This is one of the simplest and efficient classification algorithms, the main algorithm can be thought of as a graphical tree structure which incorporates various parameters and predicts the results. The decision trees use the data structure, tree to predict and give the outcome. Its fast classification of unknown records and disregard of unnecessary features resulted in it being tried for the project.
5. ADA-boosting: It is a type of boosting ensemble method. Generally, the weak classifiers used here are decision trees. The classifier is trained using weighted samples, wherein the weights are updated after each iteration to give more importance to misclassified samples and less weight to correctly classified ones.
6. KNN: This is definitely the most simple classification algorithm for images, this is because the algorithm does not learn anything and depends on the distance between different feature vectors to give the result. It finds the accurate classification result by finding the class that is most common among the k “nearest neighbours”. KNN was extremely simple for implementation given its easy to understand logic, moreover there is no time needed to train the dataset.

**Results**

| Name of the Method | Validation accuracy | Evaluation accuracy (as obtained after submission on Kaggle) |
| --- | --- | --- |
| 1. DenseNet + Multilayer Perceptron | 92.57% | 84.363% |
| 1. ResNet+MLP | 92.076% | 82% |
| 1. DenseNet + SGD | 91.77% | 79.545% |
| 1. DenseNet + SVC | 92.87% | 79.545% |
| 1. DenseNet+Bagging Classifier | 93.33% | 78% |
| 1. ResNet50 + RFC + AdaBoost | 92% | 73.18% |
| 1. ResNet50 + SVC | 89.07% | N/A |
| 1. VGG + MLP | 91.08% | 71.181 |
| 1. VGG + SVC | 91.39% | 74.27% |
| 1. ResNet50 + RFC | 84.72% | N/A |
| 1. ResNet50 + DT | 81.9% | N/A |

**Conclusion**

For the implementation of the 6-class image classifier, an emphasis was put on finding a method that combined feature extraction and classification algorithm to give highest accuracy. Of the methods that were implemented, the DenseNet + Multilayer Perceptron algorithm gave the best results (Validation accuracy = 92.57% and Evaluation accuracy = 84.363%).

**References**

1. Wang, Ruihu. (2012). AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review. Physics Procedia. 25. 800-807. 10.1016/j.phpro.2012.03.160.
2. K-Nearest Neighbor classification. (n.d.). Retrieved from <https://customers.pyimagesearch.com/lesson-sample-k-nearest-neighbor-classification/>
3. Xu, Baoxun & Ye, Yunming & Nie, Lei. (2012). An improved random forest classifier for image classification. 2012 IEEE International Conference on Information and Automation, ICIA 2012. 795-800. 10.1109/ICInfA.2012.6246927.
4. Multilayer Perceptron. (n.d.). Retrieved from <https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron>
5. Tsang, S. (2019, March 20). Review: DenseNet - Dense Convolutional Network (Image Classification). Retrieved from <https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803>
6. Bagging (Bootstrap Aggregation). (2021, July 29). Retrieved from <https://corporatefinanceinstitute.com/resources/knowledge/other/bagging-bootstrap-aggregation/>
7. (n.d.). Retrieved from <https://pages.cms.hu-berlin.de/EOL/geo_rs/S09_Image_classification2.html>
8. Raj, A. (2021, October 30). An Exhaustive Guide to Decision Tree Classification in Python 3.x. Retrieved from <https://towardsdatascience.com/an-exhaustive-guide-to-classification-using-decision-trees-8d472e77223f>
9. 30/08/2020, Enthusiast, P. V., & Varma, P. (2020, October 26). How Does PCA Dimension Reduction Work For Images? Retrieved from <https://analyticsindiamag.com/how-does-pca-dimension-reduction-work-for-images/>
10. *Support Vector Machine (SVM)*. (n.d.). Www.Tutorialspoint.Com. <https://www.tutorialspoint.com/machine_learning_with_python/classification_algorithms_support_vector_machine.html>
11. Jay, P. (2018, February 8). *Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook [Part 1]*. Medium.Com. <https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624>
12. Brownlee, J. (2016, April 25). *Boosting and AdaBoost for Machine Learning*. Machinelearningmastery.Com. <https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/>

**Appendix**

| **ResNet50+SVC**  svmclf = svm.SVC(kernel='sigmoid', verbose = True)  svmclf.fit(trainX, trainY)  predY\_svm = svmclf.predict(valX)  acc\_svm = metrics.accuracy\_score(valY, predY\_svm)  print("validation accuracy =", acc\_svm)  # 89.07… | **ResNet50+RFC + AdaBoost** RFC = RandomForestClassifier(n\_estimators = 100, n\_jobs = 10)  ARF = AdaBoostClassifier(base\_estimator = RFC, n\_estimators=100)  ARF.fit(trainX, trainY)  print("Fitting Completed")  ARF\_PY = ARF.predict(valX)  acc\_ARF = metrics.accuracy\_score(valY, ARF\_PY)  print("validation accuracy =", acc\_ARF)  # 92.0  # Test Submission Accuracy = 73.18 |
| --- | --- |
| **ResNet50+Gradient Boost with different Learning Rate**  for learning\_rate in lr\_list:  gb\_clf = GradientBoostingClassifier(n\_estimators=20, learning\_rate=learning\_rate, verbose = 1)  gb\_clf.fit(trainX, trainY)    print("Learning rate: ", learning\_rate)  print("Accuracy score (training): {0:.3f}".format(gb\_clf.score(trainX, trainY)))  print("Accuracy score (validation): {0:.3f}".format(gb\_clf.score(valX, valY)))  # lr = 0.05 --> 89.7  # lr = 0.1 --> 90.3  # lr = 0.25 --> 90.7  # lr = 0.5 --> 90.1  # lr = 1 --> 87.2 | **MobileNetV2**  base\_model = tf.keras.applications.MobileNetV2(input\_shape = (224, 224, 3), include\_top = False, weights = "imagenet") base\_model.trainable = False model = tf.keras.Sequential([base\_model, tf.keras.layers.GlobalAveragePooling2D(), tf.keras.layers.Dropout(0.2), tf.keras.layers.Dense(6, activation="softmax") ])  # 91.90 |
| **DenseNet+MLP:**  92.57% Val  84.363% after submission  *clf =neural\_network.MLPClassifier(hidden\_layer\_sizes=(300,150,100,50), random\_state=1,verbose=1, max\_iter=1000)*  *clf.fit(trainX, trainY)*  *y\_pred\_MLP = clf.predict(valX)*  *print(metrics.accuracy\_score(valY,y\_pred\_MLP))*  *predY = clf.predict(test\_feats)*  *write\_csv\_kaggle\_sub('submission\_DenseNet\_MLP.csv', test\_img\_id, predY)* | **DenseNet + SVC**  92.87% Val  79.545% after submission  *from sklearn.pipeline import make\_pipeline*  *from sklearn.svm import SVC*  *from sklearn.preprocessing import StandardScaler*  *clf = make\_pipeline(StandardScaler(), SVC(gamma='auto'))*  *clf.fit(trainX, trainY)*  *y\_pred\_SVC = clf.predict(valX)*  *print(metrics.accuracy\_score(valY,y\_pred\_SVC))* |
| **DenseNet + SGD:**  91.77% Val  79.545% after submission  *from sklearn.linear\_model import SGDClassifier*  *sgd\_clf = SGDClassifier(random\_state=42, max\_iter=1000, tol=1e-3)*  *sgd\_clf.fit(trainX,trainY)*  *y\_pred\_SGD = sgd\_clf.predict(valX)*  *print(metrics.accuracy\_score(valY,y\_pred\_SGD))*  *predY = clf.predict(test\_feats)*  *write\_csv\_kaggle\_sub('submission\_Dense\_SGD.csv', test\_img\_id, predY)* | **DenseNet + Bagging:**  93.33% Val  78% after submission  *from sklearn.svm import SVC*  *from sklearn.ensemble import BaggingClassifier*  *bagg = BaggingClassifier(base\_estimator=SVC(),n\_estimators=10, random\_state=0)*  *bagg = bagg.fit(trainX,trainY)*  *y\_pred\_bagg = bagg.predict(valX)*  *print(metrics.accuracy\_score(valY,y\_pred\_bagg))* |
| **VGG + MLP:**  91.08% Val  78.181% after submission  *from sklearn.neural\_network import MLPClassifier*  *clf = MLPClassifier(solver='adam', alpha=1e-5, hidden\_layer\_sizes=(300,150,100,50), random\_state=1,verbose=1, max\_iter=1000)*  *clf.fit(trainX, trainY)*  *y\_pred\_MLP = clf.predict(valX)*  *print(metrics.accuracy\_score(valY,y\_pred\_MLP))*  *predY = clf.predict(test\_feats)*  *write\_csv\_kaggle\_sub('submissionMLP\_VGG\_Actual.csv', test\_img\_id, predY)* | **VGG + SVC:**  91.39% Val  74.27% after submission  *from sklearn.pipeline import make\_pipeline*  *from sklearn.svm import SVC*  *from sklearn.preprocessing import StandardScaler*  *clf = make\_pipeline(StandardScaler(), SVC(gamma='auto'))*  *clf.fit(trainX, trainY)*  *y\_pred\_SVC = clf.predict(valX)*  *print(metrics.accuracy\_score(valY,y\_pred\_SVC))*  *predY = clf.predict(test\_feats)*  *write\_csv\_kaggle\_sub('submission\_SVC\_VGG.csv', test\_img\_id, predY)* |
| **ResNet50 +RFC**:  from sklearn.ensemble import RandomForestClassifier  rfc = RandomForestClassifier(max\_depth=2, random\_state=0, n\_estimators=20)  rfc.fit(trainX, trainY)  y\_pred\_rfc=rfc.predict(valX)  print(metrics.accuracy\_score(valY,y\_pred\_rfc))  84.72% | **ResNet50+DT**  from sklearn.tree import DecisionTreeClassifier  classifier = DecisionTreeClassifier()  classifier.fit(trainX, trainY)  y\_pred\_dt=classifier.predict(valX)  print(metrics.accuracy\_score(valY,y\_pred\_dt))  81.9% |
| **ResNet+GradientClassifier:**  from sklearn.ensemble import GradientBoostingClassifier  GBC = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1, max\_depth=1, random\_state=0).fit(trainX,trainY)  y\_pred\_GBC = GBC.predict(valX)  print(metrics.accuracy\_score(valY,y\_pred\_GBC))  88.5% | **ResNet+KNN**  from sklearn.neighbors import KNeighborsClassifier  neigh = KNeighborsClassifier(n\_neighbors=3)  neigh.fit(trainX,trainY)  y\_pred\_k= neigh.predict(valX)  print(metrics.accuracy\_score(valY,y\_pred\_k))  90.6% |