APPLIED MACHINE LEARNING SYSTEM ELEC0134 21/22 REPORT

SN: 21183138

ABSTRACT

In this report, a variety of machine learning algorithms are implemented to classify the type of tumor in MRI scan image. All trained learning models have undergone hyperparameter tuning and cross-validation to avoid overfitting with training set and validation set. Finally the model is evaluated with test set and the results are discussed and explained. [1]

***Index Terms*—** Image classification, data pre-processing, machine learning, deep learning, hyperparameter tuning

1. INTRODUCTION

This report is a part of assignment of Applied machine learning system ELEC0134 21/22. The main purpose of the assignment is to combine machine learning theories learned from class and programming skills learned from lab together, and apply several machine learning algorithms to complete specific tasks efficiently. For the assignment, we need to classify tumor type in MRI image with help of ML techniques.

This report is structured as follows: In the next Section several potential learning based approaches to solve the tasks are introduced and recalled. Section 3 briefly describe the models we decided to use in the assignment. In Section 4, we illustrate how these models are implemented for tasks in detail. In Section 5, experimental results is presented in order to verify performance of models. Finally, in Section 6 we conclude this report.

2. LITERATURE SURVEY

In this Section, several potential learning based approaches to solve the tasks are introduced and recalled.

Traditional image classification consists of feature extraction and classifier training. In the feature extraction, various general features are used, such as HOG for pedestrian detection [1], HAAR for face detection [2]. Then traditional models such as logistic regression [3], SVM [1] are used to train a classifier. This traditional method needs to select and design image features for different image types, which requires a lot of prior knowledge experience.

Since the advent of deep learning era, most methods used for image classification nowadays are deep learning, namely, neural networks, because the most time-consuming and laborious feature extraction of traditional methods can be omitted. Designers only need to design the network structure to allow it automatically extract image features, such as the famous network structure VGG16 [4], ResNet50 [5]. Deep learning is useful for image classification, but also has some drawbacks, such as requirement for large data amount and high computational capability.

3. DESCRIPTION OF MODELS

In this Section, the models we decided to use in the assignment will be briefly described, along with the rationale.

3.1. Logistic regression

Logistic Regression (LR) is a generalized linear model. In LR, the nonlinear sigmoid function is introduced to handle binary classification task. The essence of LR is to assume that the dataset obey the Bernoulli distribution, and then use maximum likelihood estimation (MLE) to predict task [3]. Logistic regression, as a very classic binary classification model, is popular in practical application due to its simplicity and strong interpretability. Therefore, LR is applied for binary classification task A.

3.2. Support vector machine

Support vector machines (SVM) is a binary classification model. Different from the perceptron, it constructs a linear classifier by finding a linear hyperplane that maximizes the interval in data feature space. In addition, by using kernel techniques, SVM can become a substantial non-linear classifier. Commonly used kernel are linear kernel, polynomial kernel, radial basis function kernel (RBF). SVM is a small-sample learning method with a solid theoretical foundation and can also be extended to multiple classification effectively. SVM with HOG features are widely used in image classification task nowadays [1]. So SVM is both applied for task A and B.

3.3. Neural network

The neural network (NN) builds a layer-by-layer network through many neurons, and then uses the nonlinear activation function to realize strong nonlinear fitting ability. The NN

[1]The code is provided on Google drive folder: AMLS_21-22-_SN21183138, and GitHub project: AMLS_21-22-_SN21183138.

model here specifically refers to fully connected neural network. NN has ability to learn and build nonlinear complex models, so it is widely used to infer the unknown relationship between unknown data [6]. The fully connected neural network also performs well for image classification task, where the output layer of NN for classification task is selected as softmax to be consistent with multiple label. For neural network, the main problem is overfitting, so when applying NN, regularization method such as L2 norm penalty term and dropout layer will be implemented. Neural network can be essentially seen as multi-layer version of SVM. So it's reasonable to compare model performance of them. Hence, NN is also both applied for task A and B.

3.4. Transfer learning with VGG16

In addition to above-mentioned fully connected NN, another very well-known network structure is the convolutional neural network (CNN). The basic CNN consists of three structures: convolution, activation, and pooling. The output of CNN is the specific feature space of input image, CNN can also be regarded as the feature extraction of image. Then the fully connected network is implemented to complete the classification map from the extracted image feature to the output label. A very famous CNN is VGG16 [4], which proves that increasing the depth of the network can affect the model performance. The drawback of CNN is data hungry and high computational capability.

For absence of sufficient training data, it is usually impossible to directly train the complete CNN. In this scenario, transfer learning (TL) can be implemented. Take TL with VGG16 as an example, the idea is to pre-train VGG16 on a general large-scale image data set firstly, such as ImageNet. Note that the training process may takes quite longer. Then on the basis of this pre-trained VGG16, we only train the fully connected layer of the network and remain convolutional part unchanged and locked.

Since the size of the MRI image data set is not large, this is a suitable application scenario for TL. We want to verify whether the convolution part of pre-trained VGG16 can extract MRI image features or not. Moreover, because we only train the fully connected layer, the training process can be completed effectively and within short time. Therefore, TL with VGG16 is applied for task B.

4. IMPLEMENTATION

In this section, we implement the above models for classification task A and B. Each model will be repeatedly trained and validated following the order: training, cross-validation, hyperparameter tuning. We perform 5-fold cross-validation and grid search for hyperparameter tuning and the main criteria for evaluation is accuracy and cohen kappa score. In addition, we hope training and execution of all models can maintain high efficiency, namely, one single training and validation process can be completed within short time (2 min). All classic machine learning models are implemented based on scikit-learn, and deep learning models are executed with help of Keras.

4.1. Data overview

In task A and task B, we need to use human brain scans taken by MRI to determine the brain tumor. These scans are gray scale images with 512*512 pixels. All images have been labeled by 4 tumor types, no tumor, glioma tumor, meningioma tumor and pituitary tumor. So for this data set, this is a typical image classification task.

The dataset contains 3200 MRI images, of which 3000 are used for model training and validation in this section, and 200 are used for model testing in Section 5. To be precise, the training-validation-test spilt rule is chosen as follows: 90% of 3000 MRI images is used as training set (2700 images), 10% is used as validation set (300 images) and another 200 MRI images are used as test set, as shown in Figure 1.

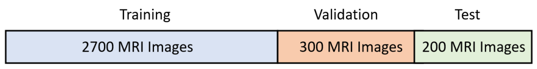


Fig. 1. Rule of train-valid-test split

4.2. Task A: Binary task

The task A is a classic binary classification task. We need to identify whether there is a tumor in the scan image. For this task, logistic regression, support vector machine and neural network are implemented. The goal is to use relatively simple machine learning algorithms to achieve excellent model performance

4.2.1. Task A: Data pre-processing

For label in task A, we use "1" and "0" to indicate and identify whether there is a tumor in MRI image or not.

For pre-processing of the MRI images, since images are susceptible to interference from noise during digitization and transmission, they need to be denoised. A common way is median filter, namely, to calculate the median of the pixels in a region [7], the size of filter kernel is chosen as 5. Then due to the characteristics of the MRI scan image, we can enhance the data by enhancing the contrast correspondingly, which can effectively improve model performance [8]. Next, we resize the images with 128$*$128 pixels and normalize it. Since gray image is essentially a matrix, we vectorize it to get a new vector data with 16384 dimensions. All above image pre-processing steps are applied for both training set and validation set, and completed by using the OpenCV library.

Next, we use this vectorized data and an experiment with SVM to illustrate importance of dimensionality reduction. We train directly a SVM with 16384-dimensional vectors of training set, and try to predict the validation set. In addition, we use PCA to reduce the dimensionality of the data, and plot cumulative variance ratio wrt. number of components. As shown in Figure 2, the first 200 principal components can retain 80% of the data variance, which indicates PCA successfully reduced the 16384-dimensional vector data into 200 dimensions, and these reduced-dimensional data are used to perform SVM again. From comparison of the results shown in the Table 1, PCA improve the accuracy of the model slightly, and effectively reduce the computation time. Note that this experiment is only to show necessity of dimensionality reduction.

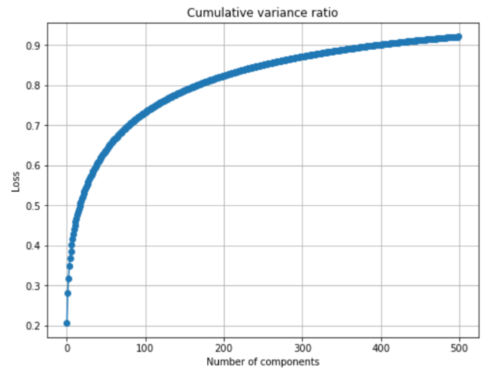


Fig. 2. Cumulative variance ratio of dataset after performing PCA for task A

Performance	Without PCA	With PCA
Accuracy	0.91	0.93
Time spent for training	>5 min	<2 s

Table 1. Result: Small experiment with SVM

The above experiment with SVM has demonstrated the importance of dimensionality reduction, hence it is reasonable to perform PCA with 200 principal components to generate the new data set. So far, the 512*512 pixels MRI images have been successfully pre-processed and reduced to 200-dimensional, which will be used for implementation of models for task A. To be precise, the new training set (2700 200-dimensional vectors) and validation set (300 200-dimensional vectors) generated from PCA will be used.

4.2.2. *Task A: Implementation of logistic regression*

In this part, the logistic regression (LR) is implemented for the binary classification task based on scikit-learn library. Firstly, a LR model with following hyperparameter is trained with training set: regularization C chosen as 0.001, maximum iteration chosen as 1000. The penalty term is the L2 norm penalty. Then the trained model is evaluated with validation set, which shows model accuracy is 0.85 [2]. For a binary classification problem, this performance is not good, and the corresponding cohen kappa score is only 0.21 (usually this value greater than 0.8 indicates a good classifier [9]). Therefore, we need to perform hyperparameter tuning to improve model.

The GridSearchCV is implemented for hyperparameter tuning, which performs a grid search on the specified hyperparameter range, then uses 5-fold cross validation to train and validate the model, finally obtains the best hyperparameter [10]. Here we choose hyperparameter range as shown in Table 2. Different value of C from small to large means that the regularization gradually increases. We hope to avoid overfitting by adjusting these two hyperparameters.

Hyperparameter	Value
Regularization parameter C	0.001, **0.01**, 1, 10
Maximum iteration	**200**, 500, 1000

Table 2. GridSearchCV: Logistic regression for task A

After GridSearchCV, the best hyperparameter of LR for task A is: C chosen as 0.01, maximum iteration chosen as 200[3], and the best accuracy score is 0.9. From this result, we conclude that the increase of C indicates much more regularization penalty on our LR model. One possible reason is that our initial C of LR model is set too small and the overfitting occurs. Furthermore, the reduction of the maximum number of iterations might correspond to the early stopping [11]. Since the training process of LR also uses the gradient descent method, early stopping is also a solution to avoid overfitting. Finally, we choose the LR model with hyperparameter from GridSearchCV and evaluate it with test set in Section 5.1.

4.2.3. *Task A: Implementation of support vector machine*

In this part, the SVM is implemented for the binary classification task based on scikit-learn library. Firstly, a basic SVM model with following hyperparameter is trained with training set: regularization C chosen as 1, kernel chosen as RBF. The penalty term of SVM is the squared L2 norm penalty. Then the trained model is also evaluated with validation set, which shows model accuracy is 0.95 and the corresponding cohen kappa score is 0.802. This performance is quite better than logistic regression, but we still need to tune hyperparameter and implement cross validation to avoid overfitting.

Again we implement GridSearchCV here for hyperparameter tuning with 5-fold cross validation. Here the hyperparameter range is chosen as shown in Table 3. Different C

[2]Detailed model performance for task A with validation set such as confusion matrix can be found in Appendix A, the same for following models.

[3]The parameters in bold in the table are the selected hyperparameter of corresponding model, the same for following models.

from small to large means different regularization penalty respectively. We try to avoid overfitting by adjusting C and hope to search a specific kernel suitable for the dataset.

Hyperparameter	Value
Kernel type	Polynomial, **RBF**, Sigmoid
Regularization parameter C	0.001, 0.01, **1**, 10

Table 3. GridSearchCV: Support vector machine for task A

After GridSearch, the best hyperparameter of SVM for task A is: C chosen as 50, kernel chosen as RBF, and the best accuracy score is 0.959. From this result, we conclude that RBF kernel brings us a better performance compared to polynomial and sigmoid, because SVM with RBF kernel has been widely verified as a very effective algorithm for classification task [12]. In addition, although the regularization penalty has become larger (larger value of C), the model performance on validation set becomes better, which also illustrates the importance of cross-validation to prevent the model from overfitting. Finally, we choose the SVM model with hyperparameter from GridSearch and evaluate it with test set in Section 5.1.

4.2.4. Task A: Implementation of neural network

In this part, the NN is implemented for the binary classification task based on Keras library, hence the suitable activation function of output layer is softmax. Firstly, we formulate the initial neural network with hyperparameter shown in “Initail” column of Table 4, but the training and validation procedure for neural network is more complex compared to LR and SVM in Section 4.2.2 and 4.2.3. This initial NN is trained with training set and evaluated with validation set in every epoch, which is summarized as following training curve as shown in Figure 3. This figure presents a classic learning process. As the epoch increases, the training loss continues to decrease, but the validation loss begins to rise after a certain period of epochs, which shows overfitting. Furthermore, the training loss converges slowly, Which indicates the fitting ability of the initial neural network is insufficient and the training epochs are too few.

Hyperparameter	Initial	**Complex**	GridSearchCV
Hidden layer (ReLU)	3	**4**	4
Hidden unit	4	**16**	16, 32
Epoch	100	**200**	200
Batch size	128	**128**	128
Dropout ratio	/	**0.2**	0.2, 0.5
Regularization C	/	**0.02**	0.02, 0.2

Table 4. Hyperparameter tuning: Neural network for task A

Therefore, we adjust the structure of the former neural network according to the loss curve, namely, to tune the

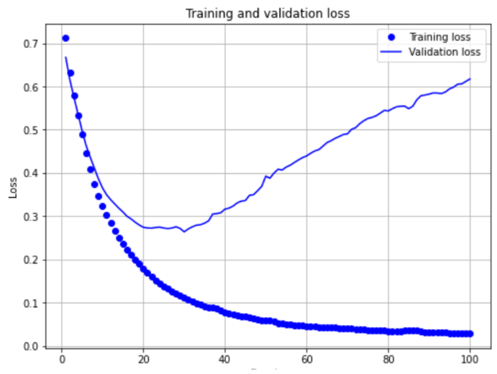


Fig. 3. Training curve: Initial neural network for task A

hyperparameters. This procedure of repeatedly tuning the hyperparameters according to the loss curve is also called trial and error. Based on Figure 3, we could modify the network deeper (more hidden layers) and bigger (more hidden units) to improve fitting ability. To avoid overfitting, the L2 regularization and dropout layer are also considered. After several hyperparameter tuning per trial and error, we formulate a complex neural network with hyperparameter shown in “Complex” column of Table 4. Then this NN is trained and evaluated, which is summarized as following training curve as shown in Figure 4. Both loss curves continue to decline and tend to stabilize, and there is always a small gap between them, which can be explained as the generalization error gap [6]. The accuracy score in validation set is 0.947, which also indicates a good binary classifier.

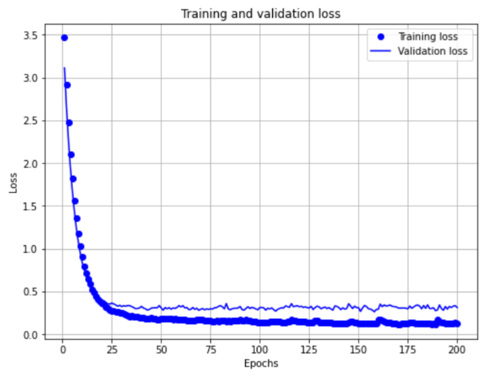


Fig. 4. Training curve: Complex neural network for task A

In the hyperparameter tuning so far, the trial and error method is applied. Now we try to implement the GridSearchCV to tune the hyperparameters based on previous result, to search for optimal L2 penalty coefficient C, dropout ratio and number of hidden units, because the first two are the main part of model regularization term and closely related to overfitting. Here the hyperparameter range is chosen as shown in “GridSearchCV” column of Table 4. From GridSearch, the best hyperparameter is: C chosen as 0.02, dropout ratio as 0.2, hidden units chosen as 32. In the next step, we retrain and validate a NN based on new hyperparameter from GridSearchCV, and the resulted accuracy score of this new NN is 0,94. Compared with the NN model from trial and error, the accuracy has dropped slightly, so we still choose the previous model (the "complex" NN) and evaluate it later with test set in Section 5.1.

4.3. Task B: Multi-class task

The task B is classic multiple classification task. We need to identify the type of tumor in the MRI image. Since LR does not perform well in task A, we do not use LR model any more in task B. Nevertheless, SVM and NN are still implemented, the training and validation procedure of them is quite familiar as shown in Section 4.2.3 and 4.2.4. So the training and validation of SVM and NN are only briefly described in this Section. Moreover, the transfer learning with VGG 16 is also implemented for task B.

4.3.1. Task B: Data pre-processing

For label file in task B, we use "0", "1", "2", "3" to identify the 4 tumor types, namely, no tumor, glioma tumor, meningioma tumor and pituitary tumor. For deep learning based model, such as NN and VGG16, one-hot encoding is implemented to match the softmax of output layer.

For pre-processing of images, firstly a experiment with SVM is done to Check whether the previous pre-processing method in Section 4.2.1 can be used task B. We train the model with dataset in task A, note that the label file should be obviously replaced by multi-classification version. A basic SVM with following hyperparameter is trained with the training set (2700 200-dimensional vectors): regularization C chosen as 1, kernel type chosen as RBF. Then the trained SVM is evaluated with validation set (300 200-dimensional vectors), which shows accuracy is only 0.856 [4]. Obviously, for the same hyperparameter, the accuracy of SVM for multi-classification has dropped by 10% compared to task A, although we could tune hyperparameter. But the essence is likely to be caused by data pre-processing. PCA is performed directly on the image and ignores lots of intrinsic feature of image itself [13].

Therefore, the pre-processing for task B is summarized as following. The first step is still filter the gray-scale images with median filter. For the second step, instead of enhancing contrast , we implement HOG descriptor to extract feature vector. HOG descriptor forms the feature vector by calculating and counting the gradient direction histogram of local area in image, and Hog descriptor with various classifiers have been widely used in classification task, especially with SVM [14]. Nevertheless, if we directly compute HOG feature with 512*512 pixels image, we will get a high dimensional feature vector, which is unfavorable for the subsequent dimensionality reduction, and it is inconsistent with our objective of completing task efficiently. So we resize the images into $128*128$ pixels and then perform HOG descriptor, which results in 34020-dimensional HOG feature vector. These above mentioned new image pre-processing steps are applied for both training set and validation set, and completed by using the OpenCV library.

For the final step, PCA is performed to reduce dimensionality, and cumulative variance ratio wrt. number of components is plotted. As shown in Figure 5, the first 400 principal components can retain 80% of the data variance, hence it is reasonable to perform PCA with 400 principal components on the HOG feature vector to generate new dataset for task B. So far, the 512*512 pixel MRI images have been reduced to 400-dimensional data, which will be used for later implementation of SVM and NN for task B. To be precise, the new training set (2700 400-dimensional vectors) and validation set (300 400-dimensional vectors) generated from PCA will be used.

Note that this pre-processing method will not be applied to TL with VGG16, the data pre-processing of VGG16 will be explained later in Section 4.3.4.

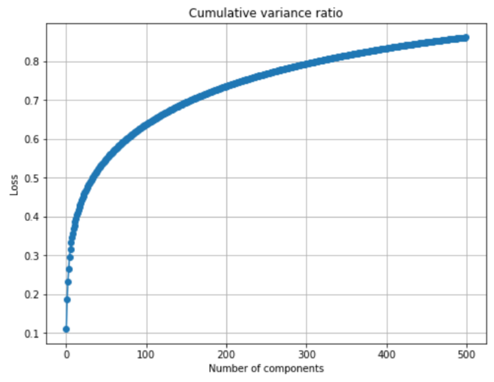


Fig. 5. Cumulative variance ratio of dataset after performing PCA for task B

4.3.2. Task B: Implementation of support vector machine

In this part, SVM is implemented for task B, and the whole procedure is quite similar to Section 4.2.3. Firstly, a basic SVM is trained as follows: regularization C chosen as 1, kernel chosen as RBF. Then the trained model shows 0.95 accuracy in validation set, and the corresponding cohen kappa score is 0.86. Again we implement GridSearchCV and the hyperparameter range is chosen as shown in Table 5. We no longer consider sigmoid kernel, because it does not perform well even in the task A. After GridSearch, the best hyperparameter of SVM for task B is: C chosen as 20, kernel chosen as RBF, and the best accuracy score is 0.93, which is way

[4]Detailed model performance for task B with validation set such as confusion matrix can be found in Appendix B

better than the basic SVM. Finally, we choose the SVM with hyperparameter from GridSearch and evaluate it with test set in Section 5.2.

Hyperparameter	Value
Kernel type	Polynomial, **RBF**
Regularization parameter C	0.1, 0.2, 1, 2, 10, **20**

Table 5. GridSearchCV: Support vector machine for task B

4.3.3. Task B: Implementation of neural network

In this part, the NN is implemented for task B, and the whole procedure is quite similar to Section 4.2.4. Firstly, we formulate the initial NN with hyperparameter shown in "Initail" column of Table 6. This initial NN is trained with training set and evaluated with validation set in every epoch, as shown in Figure 6. Then after several hyperparameter tuning per trial and error, we formulate another complex neural network with hyperparameter shown in "Complex" column of Table **??**. Then this NN is also trained and evaluated with validation set in every epoch. The accuracy score in validation set is 0.94.

Hyperparameter	Initial	Complex	**GridSearchCV**
Hidden layer (ReLU)	3	5	**5**
Hidden unit	4	256	128, **256**
Epoch	100	200	**200**
Batch size	128	128	**128**
Dropout ratio	/	0.2	0.2, **0.5**
Regularization C	/	0.002	**0.002**, 0.02

Table 6. Hyperparameter tuning: Neural network for task B

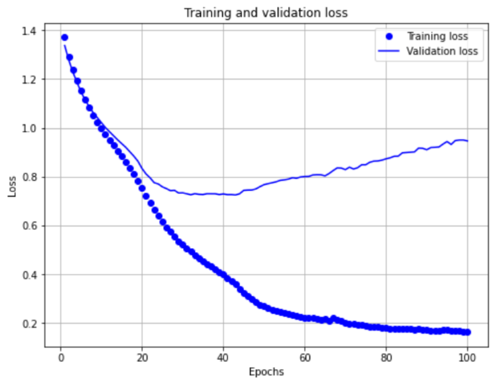


Fig. 6. Training curve: Initial neural network for task B

Moreover, we implement GridSearchCV again to tune hyperparameters based on previous result, to search for optimal L2 penalty coefficient C, dropout ratio and number of hidden units. The hyperparameter range is chosen as shown in "GridSearchCV" column of Table **??**. From GridSearch, the best hyperparameter is: C chosen as 0.002, dropout ratio as 0.5, hidden units chosen as 256. Then this NN with hyperparameter from GridSearch is also retrained and evaluated in every epoch, which is summarized as following training curve as shown in Figure 7. Both loss curves continue to decline and tend to stabilize at a small value. And the resulted accuracy score of this new NN in validation set is 0,953. Finally, compared with the model from trial and error, we choose NN with hyperparameter from GridSearch and evaluate it with test set in Section 5.2.

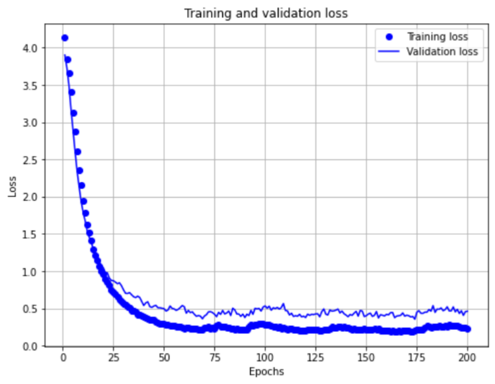


Fig. 7. Training curve: Neural network from GridSearchCV for task B

4.3.4. Task B: Implementation of transfer learning with VGG16

In this part, the TL with VGG16 is implemented for task B. Since our dataset (3000 MRI images) is and not large enough to train the entire VGG16, we use transfer learning to train only the fully connected layer of VGG16 to adapt to task B. In addition, the data pre-processing for task B is no longer applied, because the convolutional layer of pre-trained VGG16 is considered as a general feature extractor for image. We hope this convolutional layer is able to replace the previous data pre-processing, such as HOG and PCA, and achieve better feature extraction performance.

The entire training process of TL with VGG16 can be roughly divided into two steps. The first step is to compute the output of the convolutional layer of VGG16. Since we use the pre-trained convolutional layer [15], we can input the MRI image into convolutional layer and consider the output as our new dataset. Similarly, we do not directly use the full image with 515*515 pixels, because the computation burden is also too big. So we resize and normalize the images into 256*256 pixels, note that we do not implement any other pre-processing any more. To be precise, the new training set (2700 32768-dimensional vectors) and validation set (300 32768-dimensional vectors) are generated from output of convolutional layer of VGG16.

The second step is to train the fully connected layer of VGG16, which is similar to previous Section 4.2.4 and Section 4.3.3. Inspired by original structure of VGG16, we formulate the initial fully connected layer of TL with VGG16 with hyperparameter shown in "Initail" column of Table 7. We still choose the original three-layer structure, but reduce the number of hidden units in each layer by 10 times compared to structure in [4] to ensure the high efficiency and high speed of training process. For such a complex network structure, L2 regularization and dropout layer are added and applied to avoid overfitting. Here we use the GPU of Colab to accelerate the model training. This initial TL with VGG16 is trained with training set and evaluated with validation set in every epoch, which is summarized as following training curve as shown in Figure 8. Moreover, for epoch chosen as 120, this process can be finished within 2 minutes, which is also in line with our goal of training the model efficiently. Both loss curves continue to decline quickly at first and then tend to stabilize. Note that the loss curve has some small fluctuations. The accuracy score in validation set is 0.943, slightly lower than result of NN in validation set.

Now we try to implement the GridSearchCV to tune the hyperparameters based on previous result, to search for optimal L2 penalty coefficient C and dropout ratio. Here the hyperparameter range is chosen as shown in "GridSearchCV" column of Table 7. From GridSearch, the best hyperparameter is coincidentally same as initial hyperparameter, note that this does not usually occur during model training and validation. Therefore, we still choose the previous TL model and evaluate it later with test set in Section 5.2.

Hyperparameter	**Initial**	GridSearchCV
Hidden layer (ReLU)	**3**	3
Hidden unit	**500-500-100**	500-500-100
Epoch	**120**	120
Batch size	**256**	256
Dropout ratio	**0.3**	0.1, 0.3, 0.5
Regularization C	**0.0001**	0.0001, 0.0002

Table 7. Hyperparameter tuning: Transfer learning with VGG16 for task B

5. EXPERIMENTAL RESULTS AND ANALYSIS

In this Section, trained and validated models from Section 4 are evaluated with test set to show the performance of models [5]. Note that the test set here did not involve in the previous training and validation process, because we want to ensure the accuracy and validity of the results. Then the results of task A and task B are discussed and analyzed.

[5]Detailed model performance with test set such as confusion matrix can be found in Appendix C and D.

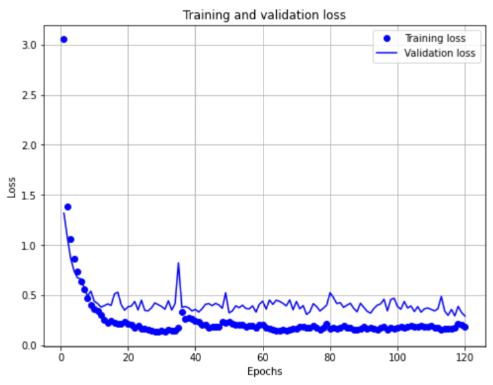


Fig. 8. Training curve: Transfer learning with VGG16 for task B

5.1. Task A: Results and analysis

Let's pre-process the test set firstly. This process is similar to steps in Section 4.2.1, but it should be noted that the PCA used here is generated from the training set and validation set, which does not contain any information of the test set.

The performance of different models for task A has been summarized in Table 8. We could conclude that among the three models, the logistic regression has the worst performance, and the error on the test set is slightly larger than validation set, which shows that overfitting still occurs. Furthermore, both SVM and neural network have very good performance for binary classification. The accuracy on the test set is above 95% and both are slightly larger than the validation set. This shows that our hyperparameter tuning benefits to avoiding overfitting.

Although the accuracy of the SVM and NN is almost the same, by observing the confusion matrix, we can summarize that the NN is more helpful for actual diagnosis. Because SVM misjudges 2 images with tumor as healthy, while NN has only 1. In real life, if a tumor is not detected, it will cause very serious and unacceptable consequences.

Model	Validation Accuracy	Test Accuracy
LR	0.899	0.875
SVM	0.959	0.965
NN	0.940	0.955

Table 8. Model accuracy for task A

5.2. Task B: Results and analysis

Let's also pre-process the test set firstly. This process is similar to steps in Section 4.3.1, but it should be noted that the PCA used here is also generated from the training set and validation set, which does not contain any information of the test set.

The performance of different models for task B has been summarized in Table 9. We could conclude that among the three models, an obvious overfitting occurs in NN, because the error of the test set is larger than validation set. Moreover, SVM with HOG and TL have good performance for multiple classification, which indicates both HOG and convolutional layer could effectively extract image feature. The both accuracy on the test set are above 95% and are slightly larger than the validation set. This shows again the necessity of our hyperparameter tuning.

However, from the perspective of training and execution efficiency, SVM with HOG is better. The training time of SVM is short, which can be completed without GPU. In contrast, although we only trained the fully connected layer of VGG16 using TL, even with GPU acceleration, the computation burden is still large. Therefore, SVM, as a very classic learning based method, can achieve high accuracy while ensuring efficiency.

Model	Validation Accuracy	Test Accuracy
SVM	0.932	0.950
NN	0.953	0.925
TL with VGG16	0.943	0.955

Table 9. Model accuracy for task B

6. CONCLUSION

In this assignment, several machine learning methods are implemented for classification task A and B. For each model, training and hyperparameter tuning with 5-fold cross validation is executed to avoid overfitting. Among these models, SVM combined with other feature extraction methods, such as PCA, HOG, has shown model performance no less than deep learning technique and could ensure the efficiency of implementation at same time.

In future, we could execute other complex deep learning models on MRI image data set, and attempt to improve the training and implementation efficiency of them.

7. REFERENCES

[1] Markus Enzweiler and Dariu M Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2179–2195, 2008.

[2] Takeshi Mita, Toshimitsu Kaneko, and Osamu Hori, "Joint haar-like features for face detection," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. IEEE, 2005, vol. 2, pp. 1619–1626.

[3] Stephan Dreiseitl and Lucila Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[4] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.

[7] David RK Brownrigg, "The weighted median filter," *Communications of the ACM*, vol. 27, no. 8, pp. 807–818, 1984.

[8] N Sasirekha and KR Kashwan, "Improved segmentation of mri brain images by denoising and contrast enhancement," *Indian Journal of Science and Technology*, vol. 8, no. 22, pp. 1, 2015.

[9] Mary L McHugh, "Interrater reliability: the kappa statistic," *Biochemia medica*, vol. 22, no. 3, pp. 276–282, 2012.

[10] Divyansh Khanna, Rohan Sahu, Veeky Baths, and Bharat Deshpande, "Comparative study of classification techniques (svm, logistic regression and neural networks) to predict the prevalence of heart disease," *International Journal of Machine Learning and Computing*, vol. 5, no. 5, pp. 414, 2015.

[11] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

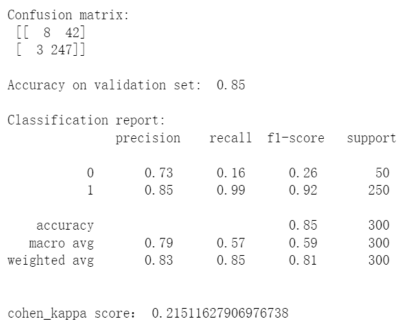
[12] Shunjie Han, Cao Qubo, and Han Meng, "Parameter selection in svm with rbf kernel function," in *World Automation Congress 2012*. IEEE, 2012, pp. 1–4.

[13] Jian Yang and Jing-yu Yang, "From image vector to matrix: A straightforward image projection technique—impca vs. pca," *Pattern Recognition*, vol. 35, no. 9, pp. 1997–1999, 2002.

[14] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Ieee, 2005, vol. 1, pp. 886–893.

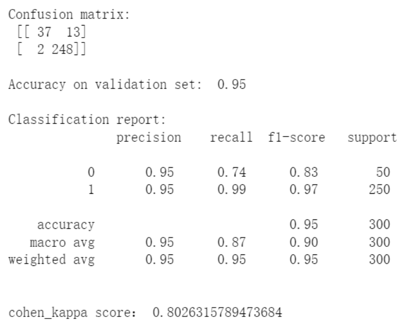
[15] "Release of pre-trained deep learning model," `https://github.com/fchollet/deep-learning-models/releases`.

A. MODEL VALIDATION OF TASK A



```
Confusion matrix:
 [[  8  42]
 [  3 247]]

Accuracy on validation set:  0.85

Classification report:
              precision    recall  f1-score   support

           0       0.73      0.16      0.26        50
           1       0.85      0.99      0.92       250

    accuracy                           0.85       300
   macro avg       0.79      0.57      0.59       300
weighted avg       0.83      0.85      0.81       300


cohen_kappa score:  0.21511627906976738
```

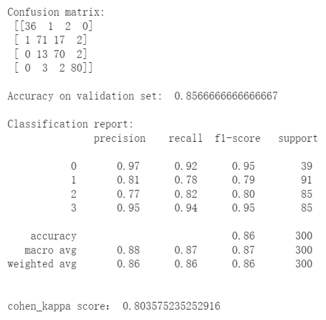
Fig. 9. Model validation: Logistic regression without hyperparameter tuning for task A



```
Confusion matrix:
 [[ 37  13]
 [  2 248]]

Accuracy on validation set:  0.95

Classification report:
              precision    recall  f1-score   support

           0       0.95      0.74      0.83        50
           1       0.95      0.99      0.97       250

    accuracy                           0.95       300
   macro avg       0.95      0.87      0.90       300
weighted avg       0.95      0.95      0.95       300

cohen_kappa score:  0.8026315789473684
```

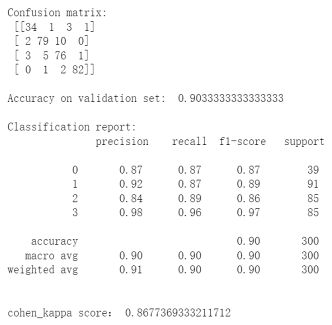
Fig. 10. Model validation: Support vector machine without hyperparameter tuning for task A

B. MODEL VALIDATION OF TASK B



```
Confusion matrix:
 [[36  1  2  0]
 [ 1 71 17  2]
 [ 0 13 70  2]
 [ 0  3  2 80]]

Accuracy on validation set:  0.8566666666666667

Classification report:
              precision    recall  f1-score   support

           0       0.97      0.92      0.95        39
           1       0.81      0.78      0.79        91
           2       0.77      0.82      0.80        85
           3       0.95      0.94      0.95        85

    accuracy                           0.86       300
   macro avg       0.88      0.87      0.87       300
weighted avg       0.86      0.86      0.86       300


cohen_kappa score:  0.803575235252916
```

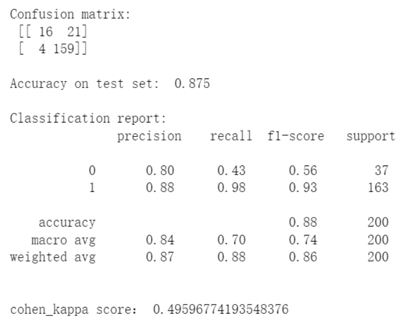
Fig. 11. Model validation: Support vector machine without HOG for task B



```
Confusion matrix:
 [[34  1  3  1]
 [ 2 79 10  0]
 [ 3  5 76  1]
 [ 0  1  2 82]]

Accuracy on validation set:  0.9033333333333333

Classification report:
              precision    recall  f1-score   support

           0       0.87      0.87      0.87        39
           1       0.92      0.87      0.89        91
           2       0.84      0.89      0.86        85
           3       0.98      0.96      0.97        85

    accuracy                           0.90       300
   macro avg       0.90      0.90      0.90       300
weighted avg       0.91      0.90      0.90       300


cohen_kappa score:  0.8677369333211712
```

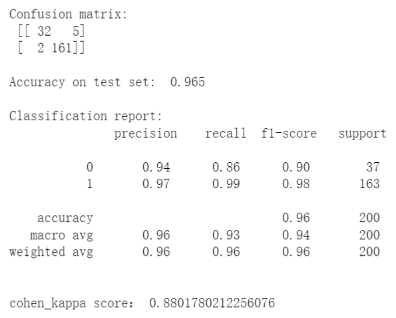
Fig. 12. Model validation: Support vector machine with HOG without hyperparameter tuning for task B

C. MODEL EVALUATION OF TASK A



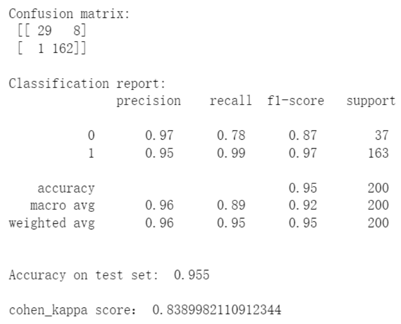
```
Confusion matrix:
 [[ 16  21]
 [  4 159]]

Accuracy on test set:  0.875

Classification report:
               precision    recall  f1-score   support

           0       0.80      0.43      0.56        37
           1       0.88      0.98      0.93       163

    accuracy                           0.88       200
   macro avg       0.84      0.70      0.74       200
weighted avg       0.87      0.88      0.86       200


cohen_kappa score:  0.49596774193548376
```

Fig. 13. Model evaluation: Logistic regression for task A



```
Confusion matrix:
 [[ 32   5]
 [  2 161]]

Accuracy on test set:  0.965

Classification report:
               precision    recall  f1-score   support

           0       0.94      0.86      0.90        37
           1       0.97      0.99      0.98       163

    accuracy                           0.96       200
   macro avg       0.96      0.93      0.94       200
weighted avg       0.96      0.96      0.96       200


cohen_kappa score:  0.8801780212256076
```

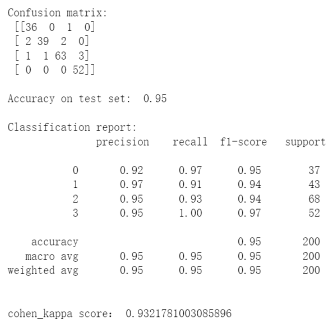
Fig. 14. Model evaluation: Support vector machine for task A



```
Confusion matrix:
 [[ 29   8]
 [  1 162]]

Classification report:
               precision    recall  f1-score   support

           0       0.97      0.78      0.87        37
           1       0.95      0.99      0.97       163

    accuracy                           0.95       200
   macro avg       0.96      0.89      0.92       200
weighted avg       0.96      0.95      0.95       200


Accuracy on test set:  0.955

cohen_kappa score:  0.8389982110912344
```

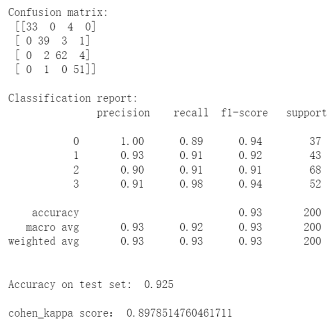
Fig. 15. Model evaluation: Neural network for task A

D. MODEL EVALUATION OF TASK B



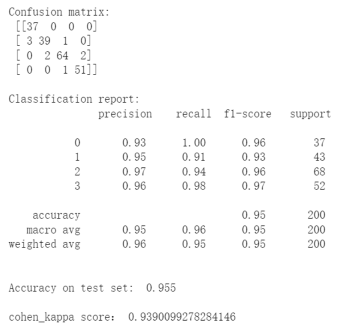
```
Confusion matrix:
 [[36  0  1  0]
 [ 2 39  2  0]
 [ 1  1 63  3]
 [ 0  0  0 52]]

Accuracy on test set:  0.95

Classification report:
               precision    recall  f1-score   support

           0       0.92      0.97      0.95        37
           1       0.97      0.91      0.94        43
           2       0.95      0.93      0.94        68
           3       0.95      1.00      0.97        52

    accuracy                           0.95       200
   macro avg       0.95      0.95      0.95       200
weighted avg       0.95      0.95      0.95       200


cohen_kappa score:  0.9321781003085896
```

Fig. 16. Model evaluation: Support vector machine for task B



```
Confusion matrix:
 [[33  0  4  0]
 [ 0 39  3  1]
 [ 0  2 62  4]
 [ 0  1  0 51]]

Classification report:
               precision    recall  f1-score   support

           0       1.00      0.89      0.94        37
           1       0.93      0.91      0.92        43
           2       0.90      0.91      0.91        68
           3       0.91      0.98      0.94        52

    accuracy                           0.93       200
   macro avg       0.93      0.92      0.93       200
weighted avg       0.93      0.93      0.93       200


Accuracy on test set:  0.925

cohen_kappa score:  0.8978514760461711
```

Fig. 17. Model evaluation: Neural network for task B



```
Confusion matrix:
 [[37  0  0  0]
 [ 3 39  1  0]
 [ 0  2 64  2]
 [ 0  0  1 51]]

Classification report:
               precision    recall  f1-score   support

           0       0.93      1.00      0.96        37
           1       0.95      0.91      0.93        43
           2       0.97      0.94      0.96        68
           3       0.96      0.98      0.97        52

    accuracy                           0.95       200
   macro avg       0.95      0.96      0.95       200
weighted avg       0.96      0.95      0.95       200


Accuracy on test set:  0.955

cohen_kappa score:  0.9390099278284146
```

Fig. 18. Model evaluation: Transfer learning with VGG16 B