Faculty of Natural Science and Engineering

Mahdi Ali Pour
Ph.D. Computer Science

CS518 - Computer Vision

Homework 3

January 2023

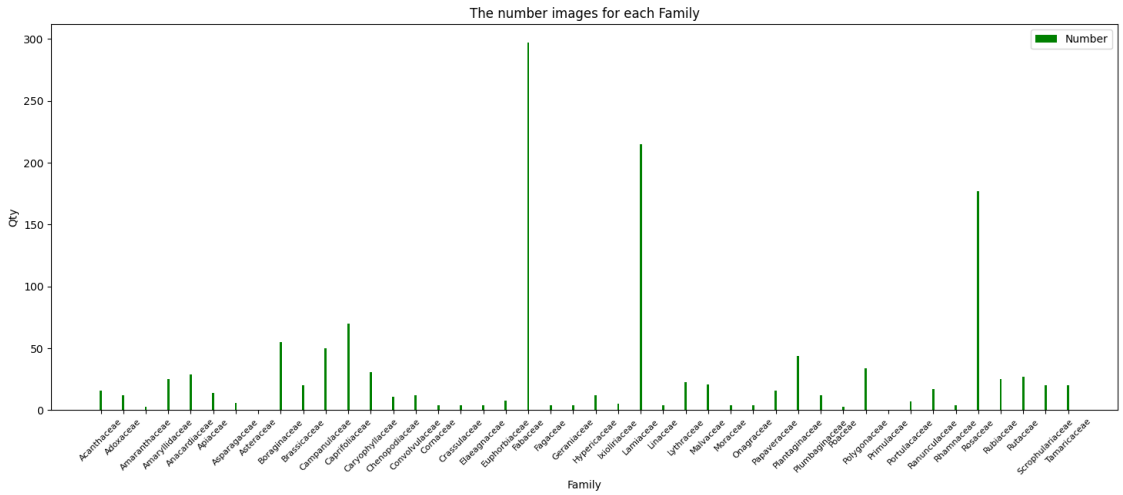

Figure 1: The number of images per their family consist of 46 labels

0.1 Task 1. Data analysis

0.1.1 Q 1.

Is this data sufficient to train a network from scratch? The number of images with respect to the number of classes (46 classes) is not sufficient. In som classes we only have 3 images!

Should we perform transfer learning? Yes, we can use a transfer learning design. It allows us to take the pre-trained weights of a neural network trained on a large dataset, and apply them to a different but related task. This can be useful in situations where we have a limited amount of labeled data for a specific task, but we have access to a large amount of labeled data for a similar task.

Is it balanced? No, the aren't. In some classes, we have more than 35 image for type and ın some other less than 5 images. With the respect to family we have worse situation.

What kind of measures should/could we take? The data is not balanced as shown in the figures 1 and 2. In another word, it is severely imbalanced dataset 1:100.

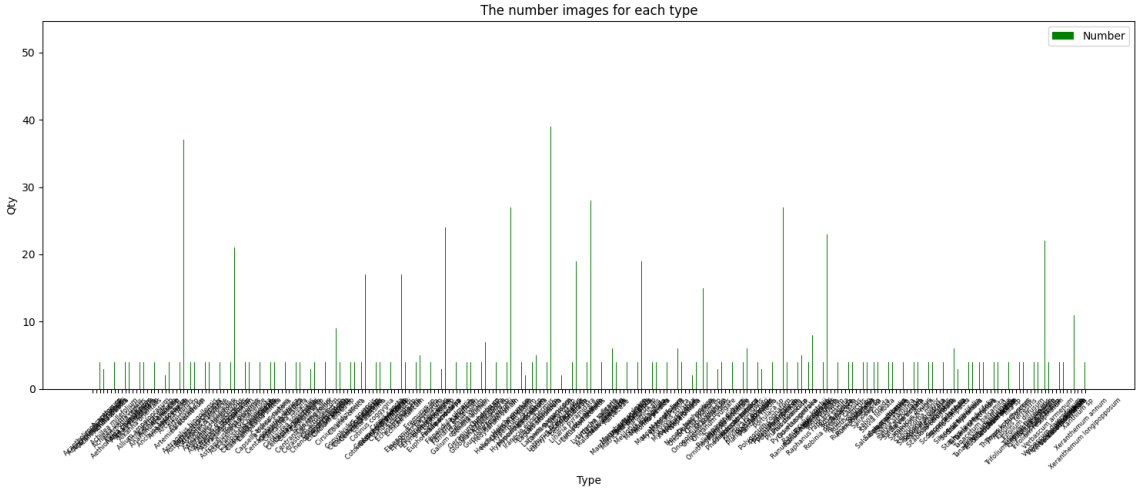

Figure 2: The number of images per their type consist of 274 labels

0.1.2 Q 2.

How should we divide this dataset into training/validation/testing subsets in order to get a split that will lead to reliable performance assessments? We need to make the data a bit balanced with some operations such as oversampling or weight balancing, then we can split to 80 and 20 for training and testing respectively. in 80% training, we split to 70, 10 for train and validation.

Should we use cross-validation? If yes, how? Yes, we can use k-fold cross-validation and then take the average of each train iteration. for the benefits of cross-validation we can name these: Stable accuracy: This will address the issue of random precision. In other words, consistent precision is possible. The model is trained on a dataset that has been divided into numerous folds. Overfitting: This stops the training data set from becoming overfitting. Validation of Model Generalization: Cross-validation provides information about how the model generalizes to unknown data sets. Validate model performance: Cross-validation allows you to precisely evaluate the prediction performance of your model. I implemented k-fold cross-validation with k=5 using the KFold class from the sklearn.model_selection module. It first defines the number of folds using the n_splits parameter of the KFold class, which in this case is set to 5. It then creates an object of the KFold class with this parameter and initializes an empty list to store the validation

scores.

What about the classes with very few samples? data can be extended by over-sampling to equal to other classes or we can balance each class's weight by assgining a number to each class according to their number of images.

0.1.3 Q 3.

What kind of a data preprocessing strategy do you recommend? Why? I used Resnet 50 which a popular pre_trained models appropriate for image classification and one of the most effective neural network architectures, which contributes to the network's ability to maintain low error rates considerably deeper within the network.

0.1.4 Q 4.

Would it make sense to use data augmentation in this scenario? If yes, what kind? Yes, since our dataset is not sufficient, we can increase the size of the dataset by augmentation. We need to be careful that the validation data should not be augmented.

This code assumes that the dataset is organized in a folder structure where each folder represents a class, and the images in each folder are the samples belonging to that class. It also uses the resample function from sklearn utils to oversample the minority class.

0.2 Task 2: Training

Network architecture To deal with the dataset I reorganized directories to get ready for family classification. I moved all types under their family. In this case, according to the word document, there were 47 families and there wasn't any image for 1 family. So the total number of families is 46. I read each class images from its directory and constract training and validation by using *ImageDataGenerator* Using ResNet50 in Keras with TensorFlow as the backend, I categorized photos, and set class weights to handle imbalanced data. Before using them to generate generators for the actual data, I first

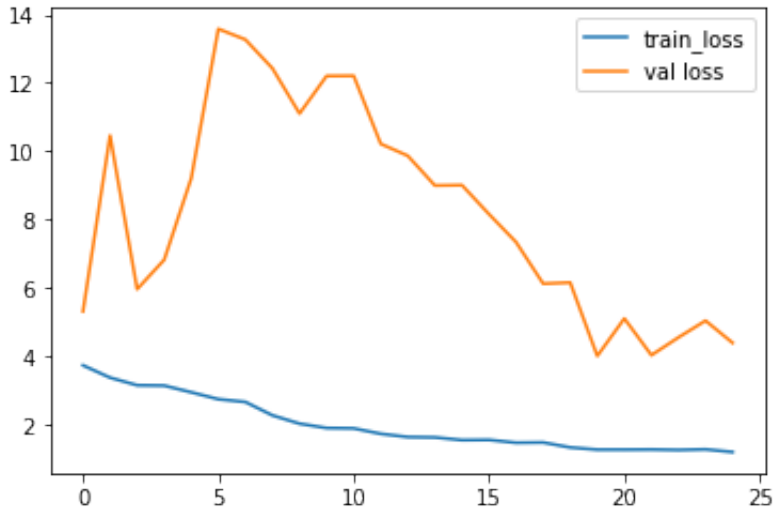

Figure 3: Training and validation loss in 1st fold for Family classification

created image data generators for the training, validation, and test data with 80/10/10 splits. The compute class weights function is then used to calculate class weights for the train data. The layers of the basic model were then frozen and I built a ResNet50 model. The base model is then added, followed by a fully connected layer with 46 classes (one for each family). In order to execute k-fold cross-validation, the model is then assembled and fitted using the SGD optimizer, categorical cross-entropy loss, and the class weights generated earlier in a for loop. I defined ReduceLROnPlateau with a minimum learning rate of 0.001 and patience of 5, then I generated an optimizer using SGD with a starting learning rate of lr=0.01 and momentum=0.9.

It then iterates over the folds using the split method of the KFold class, which returns the indices of the train and validation sets. The data is then split into train and validation sets based on these indices, and image generators are created for each set.

A new model is created for each fold and it is trained

Training/validation loss plots Unfortunately, Google Colab stopped working on the fifth fold during the training phase, so I just repeated two folds. At the end of the fourth fold, the model had 81% training accuracy and 23% validation accuracy. Because I just utilized 25 epochs, I believe it will improve with more epochs.

Augmentation I perform augmentation on training set such as: rescale, rotation, shear,

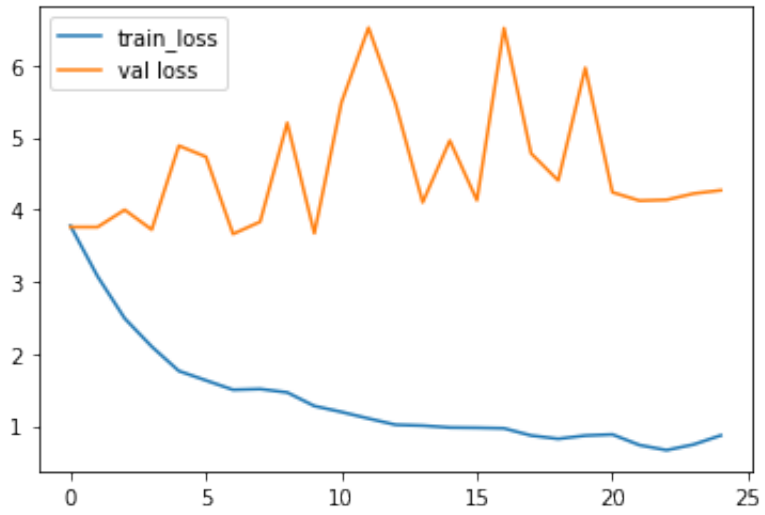

Figure 4: Training and validation loss in 2nd fold for Family classification

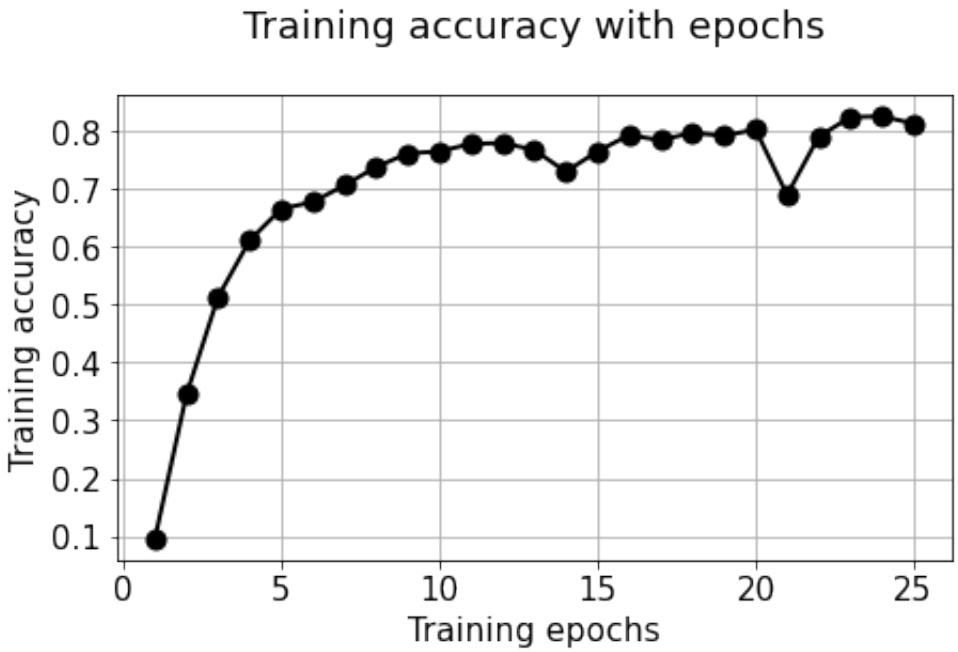

Figure 5: Training accuracy on 4th fold for Family classification

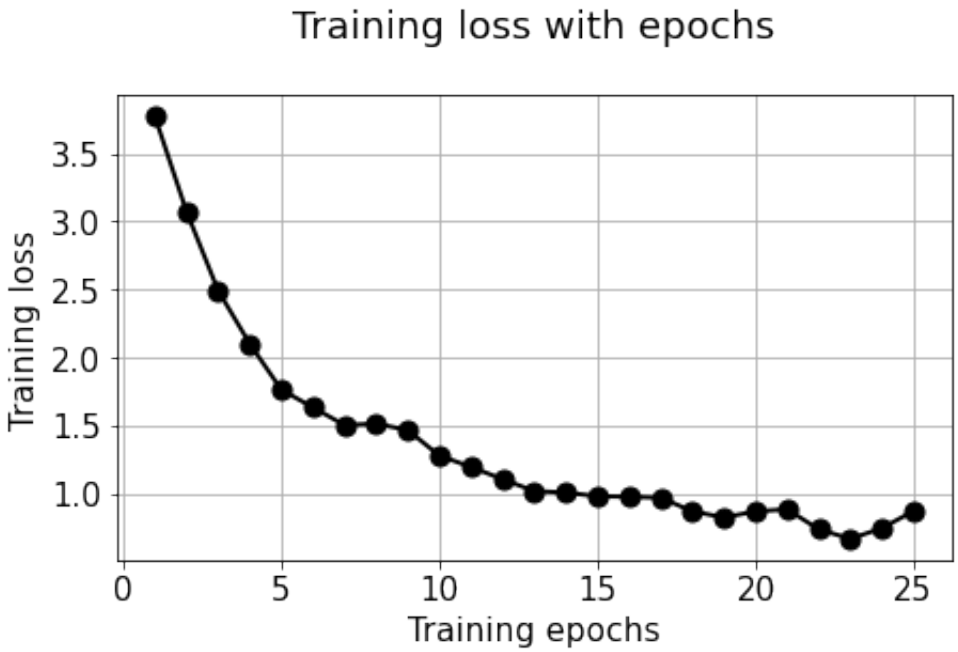

Figure 6: Training loss in 4th fold for Family classification

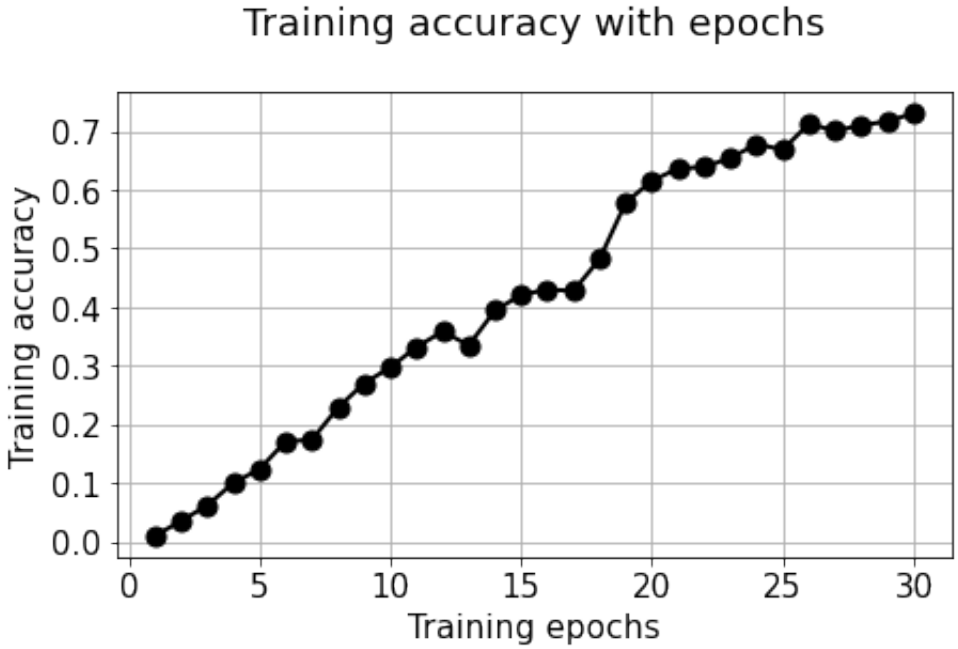

Figure 7: Training accuracy in 4th fold for Type classification

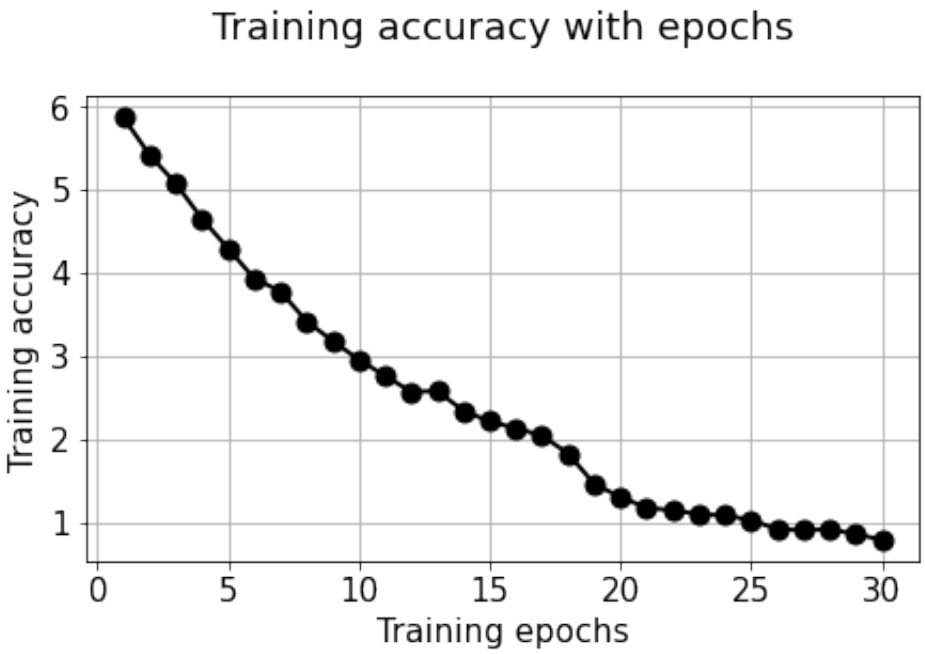

Figure 8: Training loss in 4th fold for Type classification

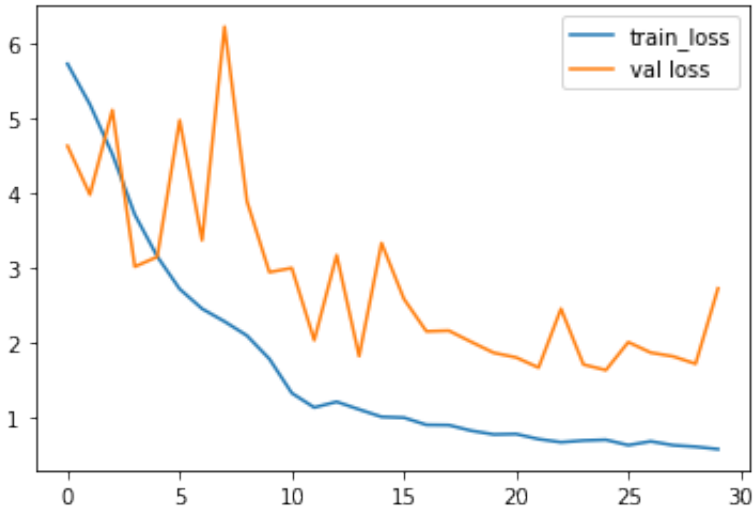

Figure 9: Training and validation loss in 1st fold for Type classification

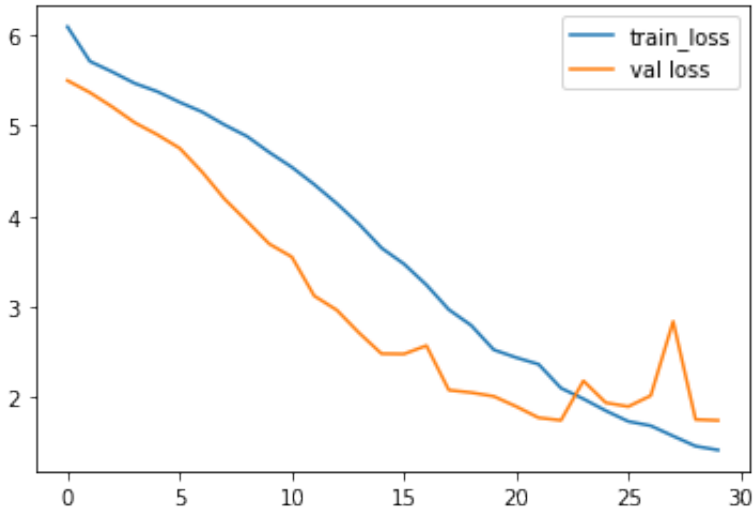

Figure 10: Training and validation loss in 2nd fold for Type classification

zoom, horizontal, and so on. It definitely affects the performance of the model since our dataset size is small, which increases the size of the dataset.

Fine-tuning Since we used the transfer learning method we need to fine-tune our model. This method involves training a pre-trained network on my task, by freezing some of the layers and training others. I used *GlobalAveragePooling2D(), Flatten, Dense 1024, Dense 512*, followed by *Relu* activation function and *Dense 46 with softmax* function.

Fold	Training accuracy	Val accuracy	Loss	Val loss
1st	0.4445	0.0375	0.9268	4.6554
2nd	0.6678	0.2062	0.4060	**3.4267**
3rd	0.7167	0.2000	0.3018	3.9176
4th	**0.8132**	**0.2375**	**0.1611**	4.2624
5th	Incomplete	-	-	-

Table 1: Family classifier accuracy and loss of training and validation in 5-fold cross-validation

Fold	Training accuracy	Val accuracy	Loss	Val loss
1st	**0.7800**	0.5030	0.5649	2.7192
2nd	0.6026	0.6213	1.4182	1.7456
3rd	0.6205	**0.7456**	**1.2622**	1.5184
4th	0.6199	0.7456	1.4889	**1.3861**
5th	Incomplete	-	-	-

Table 2: Type classifier accuracy and loss of training and validation in 5-fold cross-validation

0.3 Task 3: Inference time

Performance By observing validation accuracies we can conduct our model works on unseen data like learned data. In another word, validation accuracy and training accuracy are close. Since I repeated the training/validation process in 5 folds with different splits of the dataset so our model is generalized.

NOTE: The optimal running for training the model was at most 5-fold cross-validation (it was stopped on the fifth fold) with Google Colab. I was unable to obtain a completely trained model 5 times to get the average and mean, due to GPU runtime limitations and session disconnection after 6 or 7 hours. So I repeated the process for 2-fold cross-validation and then saved models. For evaluating on test dataset, results shown in the table below are derived from the final model that was trained in 2-fold cross-validation.

Test result The table below shows evaluating results for type and family classification models. I expected to get higher accuracy on Family classification since the number of labels is lower, but I got lower results. I used the same model as the type classifier and tried to test different parameters in the optimizer, FC layers, and splitting data but I couldn't find the problem. I am going to try different data splitting at an appropriate

Model name	Test Accuracy
Type classification	0.74
Family classification	0.23

Table 3: Type classifier accuracy and loss of training and validation in 5-fold cross-validation

time in the future to improve it.