Final project of Data Science

Classification of incident related image using machine learning (CIRI) Primary Topic: CV&IC & DM, Course: 2023-2A – Group: 32 – Submission Date: 2023-04-23

Sara Miotto
University of Twente
s.miotto@student.utwente.nl

Koen Reefman
University of Twente
k.h.a.reefman@student.utwente.nl

ABSTRACT

The immediate detection of natural disasters and other catastrophes that require human assistance is crucial for disaster relief agencies. Rapid acquisition and processing of information is therefore essential for ensuring a swift human response. Currently, this analysis is done manually, resulting in a very inefficient process. Computer Vision allows for this process to be automated, emerging as a significant gain in both time and accuracy. With this project, the aim is to present and compare different frameworks for image classification of incident related images.

KEYWORDS

Computer Vision, Deep-learning, Neural Networks, Classification, Performances

1 INTRODUCTION

Natural catastrophes can occur at any time and have terrible effects. A rapid and efficient response to them is crucial to reduce their impact as much as possible. Image classification can help by giving important information about the resources needed for disaster relief. However, traditional human-based image classification techniques are quite inefficient. For this reason, machine learning and deep learning approaches for computer vision have been increasingly utilized. Large scale datasets have been created to generate models which can automatically predict these incident related images.
The main goal here is to assess how well different frameworks handle this problem. Specifically, the research question addressed is: "What is the optimal classification system for accurately categorizing images of accidents and natural disasters?" In order to answer, some sub-questions will also be explored:

- Can transfer learning be used to enhance the performance of deep learning models for image classification tasks, and if so, what approach is most effective?
- Since the dataset used is unbalanced, how can this difficulty be addressed in evaluating the performances of the different frameworks?

A machine learning strategy, notably the K-nearest neighbors method, was initially used. However, alternative neural network models, such as Resnet, Alexnet, VGG, Squeezenet, and Densenet, were investigated in search of superior results.
The structure of the paper is the following: "Background and Related Work" offers a thorough review of the required knowledge for a better understanding of the experiments. In "Approach", the different frameworks implemented are discussed in detail. The "Experiments" section illustrates the criteria employed for assessing the performances. "Discussion" shows a reflecting evaluation of the results. Lastly, "Conclusion" provides a description of the current situation and any potential future directions.

2 BACKGROUND AND/OR RELATED WORK

Predicting the time and the location of catastrophic events can be challenging due to their unexpected nature, which may also be influenced by factors such as weather conditions, geological incidents, and other natural phenomena. Therefore, it is crucial to rely on specific tools and techniques, including cutting-edge monitoring and detecting technology.
Social media can be very useful to quickly spread real-time information. In fact, many business companies, such as the NSW Transport Management Centre, have noticed this and set goals to incorporate social media data into existing systems. In the specific scenario of transport monitoring systems, the inclusion of social media data can help to identify possible issues and respond to incidents more quickly, improving traffic flow and enhancing road user safety ([3]). Yet, finding images that are pertinent to humanitarian needs can be difficult and time-consuming because social media streams are typically clogged with meaningless content. Moreover, a substantial amount of labeled training data is required to train a deep learning model robust enough for automatic image filtering. The multi-label *Incidents1M Dataset*, a sizable collection of 1,787,154 photos, with 43 incident categories and 49 place categories, is presented by the researchers as a solution to this problem ([3]).
To prepare and align the knowledge of the group, preliminary knowledge gathering from academic articles was conducted. The group did literature reviews on the main topics of interest for the project, thus ML classification methods, insights of the *Incidents1M Dataset*, deep learning frameworks, regularization and optimization techniques ([2]) as well as research regarding the state of the art methodology in computer vision so far([5]). Each of the topics is closely related to the RQs and will be discussed below.
Initially, a k-NN model was trained ([6]). However, it was quickly discovered that a deep learning approach was far more effective in terms of accuracy. Thanks to powerful GPU computers and enhanced regularization techniques, CNNs, a specific type of multilayer neural network that simulates the functioning of the optical system in live creatures, have attained extraordinary performance in image classification ([1]).
2012 saw the introduction of AlexNet, a CNN architecture with eight layers. It became famous for the use of ReLU, multiple GPUs, data augmentation, and dropout, which were very innovative at the time.
In 2014, VGG was created to address the depth problem. In fact, training very deep neural network could result in both poor performance and vanishing gradients. VGG addressed this problem by utilizing tiny convolutional filters ($3x3$), allowing for the stacking

of more layers while yet keeping the amount of parameters under control. In order to lower the spatial dimensions of the feature maps and, consequently, the number of parameters in the following layers, it also implemented max pooling layers after a few convolutional layers ([4]).
Squeezenet is a state-of-the-art CNN architecture that was introduced in 2016 with the aim of obtaining high accuracy in image classification tasks while requiring fewer parameters and processing resources in contrast to earlier models with comparable performance. In order to do this, data is processed first through "squeeze" levels, which reduce the number of input channels, and then through "expand" layers, which increase the output channels.
Densenet121 was also created in 2016 and it addresses the vanishing gradient problem. It contains 121 layers, which allows for feature reuse, and uses dense blocks lowering the amount of required training parameters. In a dense block, each layer receives input from all layers that came before it and delivers its feature maps to the layers that follow.

3 APPROACH

As mentioned in the previous chapter, several different models were tested to obtain the best classification possible. From 'simple' algorithms like the k-NN algorithm to more advanced CNN models. Special focus was given to the Resnet18 model, which performed slightly better than other models used during transfer learning. For this reason, it was decided to implement it from scratch with some additional regularization and optimization techniques. The following subsections will explain each framework in more details.

3.1 K-nearest neighbors

The k-NN algorithm is a popular ML classification method based on a multi-step process. The first thing to do is determining the value of K, which represents the number of neighbours. Next, the Euclidean distance between the point meant to be classified and the others within the dataset is computed. Thirdly, based on proximity, the estimated distances are used to identify the K neighbors. So, the data points belonging to certain labels are counted, and then the class with highest amount of neighbors is selected.

3.2 Transfer learning

Transfer learning is another method for creating image classification models. Here the frameworks mentioned in the previous chapter were tested (Resnet18, AlexNet, VGG11, Squeezenet, and Densenet121). All of them were pre-trained on the *imagenet1k* dataset, which contains millions of labeled images across 1000 different categories. In order to use them with *Incidents1M Dataset*, the models were reshaped to have 12 outputs, corresponding to the number of classes.

3.3 Resnet 18 from scratch

Why is Resnet18 particularly convenient? The issue of vanishing gradients occurs frequently during training with back propagation and gradient descent. What happens is that the gradient, while flowing through the layers of the neural network, is subject to a cascade of arithmetic operations which can potentially reduce its magnitude to such a low value that it could affect negatively the performance of the network or even saturate the learning process. One solution to this problem is the use of residual blocks. These are made of so-called skip connections, which connect the outputs of one layer to succeeding levels by skipping some intermediary ones and in this way not learning certain connections (such as identity). During this connection-skipping phase, the network is compacted, accelerating learning. After being compressed, the layers expand, enabling the rest of the nodes of the NN to explore a bigger feature area. So, the gradient can move more fluidly while still maintaining rather significant values. Additionally, residual blocks introduce the idea of learning residuals, which entails going beyond only the mapping function to try to predict the residues resulting by the difference between the input and output and allowing the network to learn even more complex functions.
Resnets are made of residual blocks and are categorized as directed acyclic graphs network because of their complex layered structure, predefined input size of $224x224x3$, and the fact that layers take input from and give output to several other layers. Resnet18 is a network that has gained a lot of attention because of its ease of use and superior performance on a variety of applications. In contrast to other deep learning systems that have hundreds or even thousands of layers, it is quite simple. As a result, training becomes simpler and faster.
The internal layout consists of 18 layers, of which 16 convolutional and 2 fully connected at the end. The first layer has a dimension of $7x7$ with a stride of 2, which reduces the size of the input image in half. The succeeding one is max-pooling with a $3x3$ filter and a stride of 2. Four groups of convolutional blocks, each with two to three convolutional layers, receive the output of this layer. The final layer in each of these groups incorporates batch normalization, average pooling layer and ReLu as activation function. The last layer of this network has an output size equal to the 12 classes of the dataset.
In order to prevent the problem of overfitting, four regularization techniques have been implemented: L2, batch normalization, early stopping and dropout. An explanation of each of these techniques can be found in appendix A).

4 EXPERIMENTS

To fairly compare the performances, the aforementioned architectures were trained all with the same parameters. This means for the same amount of epochs, with early stopping enabled and with the same optimizer function (Stochastic Gradient Descent). The performances of the frameworks are reported in the following subsections according to multiple criteria:

- Confusion Matrix: a graphical representation of the model's true positives, false positives, true negatives, and false negatives classified items;
- Accuracy: a widely-used metric which computes the proportion of accurate predictions to all of the model's predictions;
- Weighted Accuracy: a method calculating the average accuracy by taking into consideration the number of samples in each class. This statistic gives more importance to classes with more samples, because misclassifying a sample in a larger class can significantly lower the model's overall accuracy;

- Precision: ratio of the number of correct predictions of an event to the total number of times the model predicts it;
- Recall: ratio of correct predictions for a class out of the total number of cases in which it actually occurs;
- F1 score: harmonic arithmetic mean of precision and recall;
- K-fold Cross Validation: a way of measuring how effectively a model generalizes to new data. The model is trained and tested k times overall, with the dataset partitioned into k equal-sized sections. One of the k subsets is used as a validation set to assess the model's performance after each iteration, while the remaining k-1 subsets are used for training. The results from each iteration are then combined to produce a more precise estimation of the model's overall performance;
- t-SNE: a non-linear, unsupervised method that is mostly employed for data exploration and high-dimensional data visualization allowing to get a sense of how the data is organized in a high-dimensional space. The result is a scatter plot of the data points in low-dimensional space.

4.1 Dataset

Creating deep learning models for automatic image filtering requires a lot of labeled image training. The *Incidents1M Dataset* is such a dataset. There are many pictures in this enormous collection, both in the incident and place categories. The event categories were created by merging visually similar categories and eliminating the improbable ones from a lengthy list of 233 incident classes that were discovered online. 43 event categories were created as a result, while the 49 location ones were generated by combining the 118 outdoor in the Places Dataset. Over 1.78 million photos total in the dataset, 977, 088 of which are incident with at least one incident category and 810, 066 incidents without an incident category but with at least one class-negative label. Similarly, 764, 124 photographs in the place categories are class-positive, while 1, 023, 030 images are class-negative. Since different labels can co-occur, the dataset is also multi-label. It also shows data on how frequently incidents happen in relation to other incidents or locations. In this study, a subset of the *Incidents1M Dataset* will be used to conduct the experiments. To facilitate the creation of accurate models, the dataset has been divided into three subsets: training (80%), validation (10%), and testing (10%). Table 1 presents statistics about the subset used in this study, it contains information regarding the distribution of samples over the different classes and the percentage of images for each of them. Figure 1 shows one example image per class for all classes in the dataset.

4.2 K-NN

In order to set the value of K, first the grid search hyper-parameter optimization technique was implemented. However, due to limitations imposed by Google Colab, a heuristic approach had to be used instead. For this, multiple values for K were tested. Table 2 shows the accuracy values achieved for each K, which clearly shows that these results are sub-optimal, making it clear that this approach is not particularly suitable for the goal of this paper.

Table 1: dataset distribution

Class	Samples	Percentage
0	569	12.02%
1	228	4.82%
2	249	5.26%
3	497	10.50%
4	393	8.30%
5	708	14.96%
6	615	12.99%
7	231	4.88%
8	248	5.24%
9	249	5.26%
10	498	10.5%
11	249	5.26%

Table 2: k-NN performances

	K=1	K=5	K=10	K=15	K=20	K=25	K=30	K=35	K=40
Accuracy	0.24	0.27	0.28	0.30	0.29	0.29	0.28	0.29	0.29
Precision	0.26	0.30	0.29	0.27	0.24	0.24	0.25	0.26	0.26
Recall	0.24	0.27	0.28	0.31	0.29	0.29	0.28	0.29	0.29
f1	0.22	0.25	0.25	0.26	0.24	0.24	0.23	0.24	0.24

Table 3: Performances of the different frameworks pretrained

	Resnet	Alexnet	Densenet	Squeezenet	VGG
Accuracy	0.75	0.66	0.68	0.70	0.74
Weighted Accuracy	0.80	0.70	0.78	0.75	0.78
Precision	0.79	0.72	0.76	0.71	0.76
Recall	0.75	0.66	0.68	0.70	0.74
f1	0.76	0.65	0.68	0.69	0.74

4.3 Transfer learning

As mentioned before, in search of better accuracies, multiple transfer learning models were tested and their accuracies compared. Table 3 shows the results of training these different models. It has emerged that Resnet performed best in transfer learning, although the difference with the other frameworks is not that big. The corresponding confusion matrices can be found in appendix A.3.

It should be pointed out that all models stopped before 50 epochs due to early stopping. The Resnet model trained for two epochs, after which early stopping got triggered; Alexnet trained for two epochs; Densenet just for one; Squeezenet trained for four epochs; And lastly, VGG for two epochs.
Table 5 presents the predictions of various transfer learning models on image 14, which was accurately classified every time. In contrast, 6 depicts the predictions made by the same models on image 15, which was mostly misclassified. Notably, the Squeezenet model did correctly classify this image. A graphical representation of predicted labels can be found in the form of t-SNE graphs, which can be found in appendix A.4
As mentioned in a previous chapter, k-fold cross verification has

Figure 1: One image per class

Figure 2: airplane accident

Figure 3: bicycle accident

Figure 4: car accident

Figure 5: collapsed

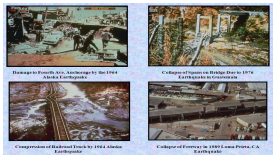

Figure 6: earthquake

Figure 7: flooded

Figure 8: ice storm

Figure 9: nuclear explosion

Figure 10: oil spill

Figure 11: tornado

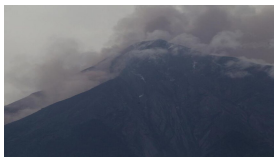

Figure 12: volcanic eruption

Figure 13: wildfire

Table 4: Accuracies of different models with k-fold cross verification

	Resnet	Alexnet	Densenet	Squeezenet	VGG
Fold 0	0.8326	0.7129	0.7837	0.7605	0.8027
Fold 1	0.8217	0.7129	0.7810	0.7605	0.8013
Fold 2	0.8204	0.7129	0.7918	0.7605	0.7972
Fold 3	0.8272	0.7129	0.7891	0.7605	0.8054
Fold 4	0.8122	0.7129	0.7823	0.7605	0.8000
Avg. accuracy	0.8229	0.7129	0.7856	0.7605	0.8014

Table 5: (Correct) predictions of different transfer learning models of image 14

Actual label	Airplane accident
Resnet	Airplane accident
Alexnet	Airplane accident
Densenet	Airplane accident
Squeezenet	Airplane accident
VGG	Airplane accident

also been used to measure how effectively each model generalizes to new data. Table 4 shows the results for each model with a K of 5.

4.4 Resnet 18 from scratch

Resnet18 from scratch was trained for just five epochs; Again, due to limitations set by Google Colab. Consequently, the accuracy

Table 6: (Incorrect) predictions of different transfer learning models of image 15

Actual label	Collapsed
Resnet	Earthquake
Alexnet	Earthquake
Densenet	Airplane accident
Squeezenet	Collapsed
VGG	Earthquake

Figure 14: Correctly classified image

obtained by the model was 0.17. Chapter 5 further discusses this result.

5 DISCUSSION

As mentioned in chapter 4.1, only a small subset of the *incidents1M dataset* was used in this project. This dataset is unbalanced, as it can be seen in the table, where it is evident that certain labels have more samples than others (for example class 5 has 708 samples, while

Figure 15: Incorrectly classified image

class 1 228). In addition to the dataset's imbalance issue, corrupt photos were also present in the dataset. The way these images were filtered out was by using the *is-valid-file* [1] parameter of PyTorch's ImageFolder, by just trying to open the image file for a given path and returning True if it succeeds and false otherwise.

The previous section has examined the outcomes of different frameworks to classify images into their respective categories of accidents and natural disasters, each yielding varying degrees of success. As stated in chapter 4.4, the training of our own Resnet18 model did not yield the desired outcomes. The primary factor of this subpar performance of the Resnet model was hardware limitations. In order to train the models, Google Colab was utilized, which presented several restrictions concerning training capabilities. For instance, the "I am human" prompt appeared at regular intervals, preventing overnight model training. Additionally, there were constraints on the duration of GPU usage, making it quite impossible to train models for an extended period of time, since training on a CPU takes significantly longer.

In terms of training time and model accuracy, transfer learning has proved to be the most promising one in this research. Transfer learning vastly outperformed the other approaches in terms of accuracy. Some transfer learning models were able to reach accuracy rates of around 80%, with Resnet slightly outperforming all other trained models. It should be emphasized that although the transfer learning models' training times were considerably lower than those for models created from scratch, they still needed a lot of time and were only trained for a limited number of epochs. Even though Resnet now outperforms all other models, we cannot make any firm conclusions without more testing due to the small number of epochs used to train each model, which may have had an impact on the accuracies. At this point, it can not be concluded that other models would not outperform the Resnet model if they were trained for an extended period of time. These pre-trained frameworks have already mastered the ability to recognize the most crucial elements in photos, making this method frequently more effective than training a new model from scratch.

As far as future work is concerned, the different models should be trained for a larger amount of time on more powerful machines, without the various constraints imposed by Google Colab. Another interesting approach would be to use Vision Transformers (ViTs), which seem to be the state of the art in the computer vision field.

In recent years, Vision Transformers (ViTs) have emerged as a viable Convolutional Neural Networks (CNNs) substitute for computer vision tasks requiring image recognition. ViTs have shown considerable gains in processing speed and accuracy when compared to CNNs. ViTs use the self-attention mechanism, a key component of the Transformer design, to find long-range dependencies and contextual information in input data. This technique enables ViT models to concentrate on different parts of the input data according to their relevance to the task at hand. When more research is done, it will be interesting to see how ViTs may be used for different computer vision tasks and what their ultimate potential is.

[1] https://pytorch.org/vision/main/generated/torchvision.datasets.ImageFolder.html

6 SUMMARY

Large-scale image classification jobs frequently encounter the problem of class imbalance, where the classifier may prefer the majority class and perform poorly on the minority classes. There are several ways to tackle class imbalance in image classification jobs using deep learning algorithms. In order to deal with the class imbalance, particular evaluation indicators were taken into account. Conventional measurements like accuracy might not give a clear picture of the model's performance, as they take into account both the true positive and false positive rates of the model. For an imbalanced dataset, metrics like weighted accuracy, recall, F1-score, and the area under the receiver operating characteristic curve (AUC-ROC) are more applicable.

This study has shown that transfer learning can really help in improving accuracies of predictions of incident related images, with the Resnet model proving to be the most capable under the current circumstances.

After comparing multiple different approaches to the problem, no clear winner emerged. However, it was determined that certain deep learning frameworks fared better than conventional k-NN. Considering the training duration, all of the evaluated transfer learning models generally attained good levels of accuracy. In this particular challenge, the Resnet model performed the best. True also that it was subjected to the regularization approaches previously described, which undoubtedly was useful.

In order to increase the accuracies of the approaches used in this paper, the models should be trained on better hardware, without the constraints imposed by Google Colab. This could lead to a different model emerging as the winner. Another interesting next step would be to explore vision transformers.

REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[2] Pushparaja Murugan and Shanmugasundaram Durairaj. 2017. Regularization and Optimization strategies in Deep Convolutional Neural Network. *CoRR* abs/1712.04711 (2017). arXiv:1712.04711 http://arxiv.org/abs/1712.04711

[3] Hoang Nguyen, Wei Liu, Paul Rivera, and Fang Chen. 2016. TrafficWatch: Real-Time Traffic Incident Detection and Monitoring Using Social Media. In *Advances in Knowledge Discovery and Data Mining*, James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang (Eds.). Springer International Publishing, Cham, 540–551.

[4] Srikanth Tammina. 2019. Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *International Journal of Scientific and Research Publications (IJSRP)* 9, 10 (2019), 9420. https://doi.org/10.29322/IJSRP.9.10.2019.p9420

[5] Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. 2022. A-ViT: Adaptive Tokens for Efficient Vision Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10809–10818.

[6] Shichao Zhang and Jiaye Li. 2020. KNN Classification with One-step Computation. *CoRR* abs/2012.06047 (2020). arXiv:2012.06047 https://arxiv.org/abs/2012.06047

A APPENDIX

A.1 Regularization techniques

In this subsection an explanation of the different regularization techniques implemented is provided.

A.1.1 L2-Regularization. L2-regularization, or weight decay, was applied to convolutional layers. It aims at limiting the model capacity by adding the squared magnitude of the coefficient as an additional term to the loss function.
The loss function takes on the following form: $\theta = \sum L(f(x_i, \theta), y_i) + \frac{\lambda}{2} \sum ||w_l||^2$
The L2-regularization can pass inside the gradient descent update rule:
$(w_l)^{(t+1)} = w_l^{(t)} - \epsilon_t(\nabla wL + \lambda w_l^{(t)})$
$w_l^{(t+1)} = (1 - \epsilon_t \lambda) w_l^{(t)} - \epsilon_t \nabla_w L$

A.1.2 Batch Normalization. A deep network can be parameterized again using the batch normalization technique. The distribution of layer inputs actually varies as learning occurs as a result of parameter modifications. Internal covariate shift is a phenomena that can cause most inputs to be in a nonlinear regime and delay learning. This can be resolved via batch normalization, which determines the mean and standard deviation of every layer's activation (see figure 16).

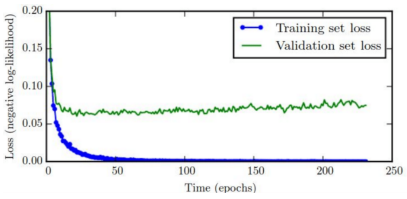

Figure 16: Batch Normalization

A.1.3 Early Stopping. Early stopping means starting with small weights and stopping the learning before it overfits. As stopping criterion a monitor on the accuracy computed on the validation set has been set. (see figure 17)

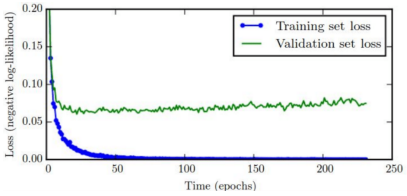

Figure 17: Early Stopping

A.1.4 Dropout. A predetermined percentage of the neurons in a particular layer are intended to be randomly dropped out (set to zero) with each training iteration. This drives the network to learn more robust features by preventing it from being unduly dependent on a single set of features, which can lead to overfitting. During training, a subset of neurons is randomly selected to remain active, while the others are deactivated. The probability of each neuron being active is frequently set to a network hyperparameter, which is typically in the range of 0.2 to 0.5. Then, during testing, all neurons are used, but their outputs are adjusted according to the chance that they were active during training. This ensures that the network produces consistent results when tested, even while some neurons are arbitrarily destroyed during training.

A.2 Backpropagation

The process of backpropagation is used to calculate gradients, where the gradient is a vector of partial derivatives with respect to all of the weights' coordinates. Each partial derivative calculates the rate of change of the loss in a single direction. Finding the model parameters that correspond to the best match between projected and actual outputs is the goal of gradient descent (see figure 18).

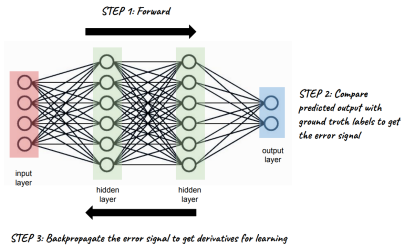

Figure 18: Backpropagation

A.3 Confusion matrices for different transfer learning models

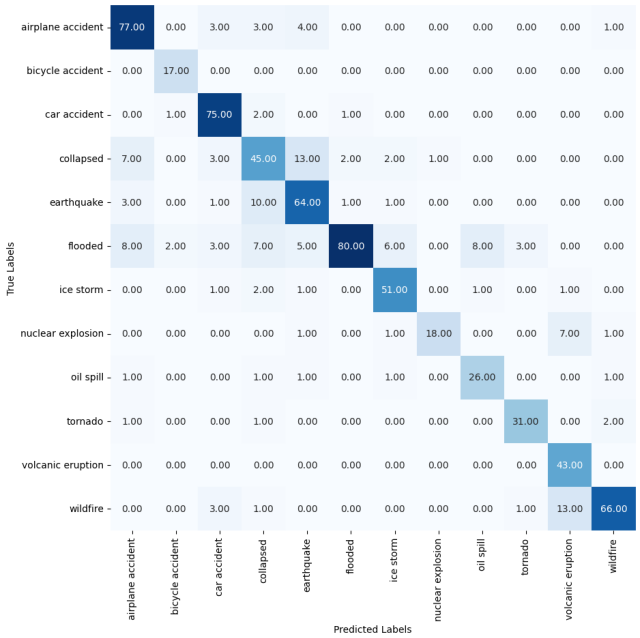

Figure 19: Confusion matrix for Resnet

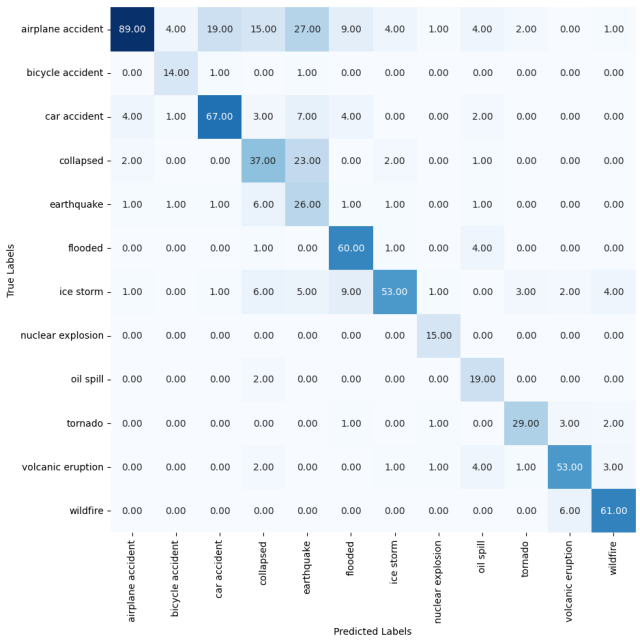

Figure 20: Confusion matrix for alexnet

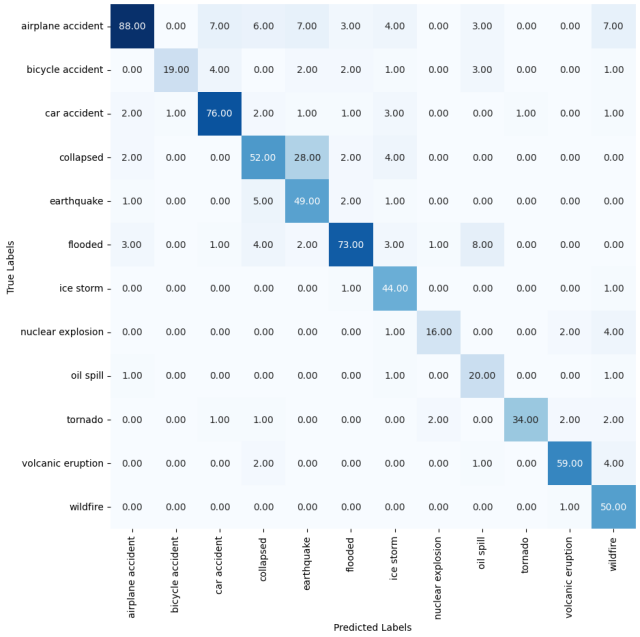

Figure 21: Confusion matrix for densenet

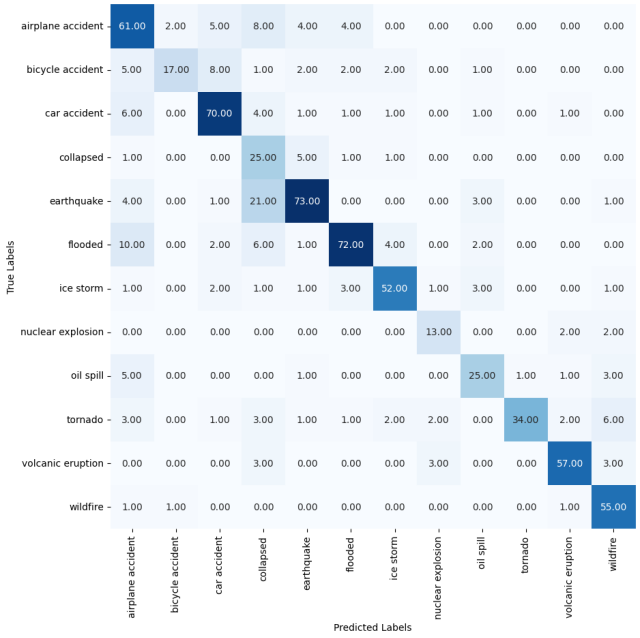

Figure 22: Confusion matrix for squeezenet

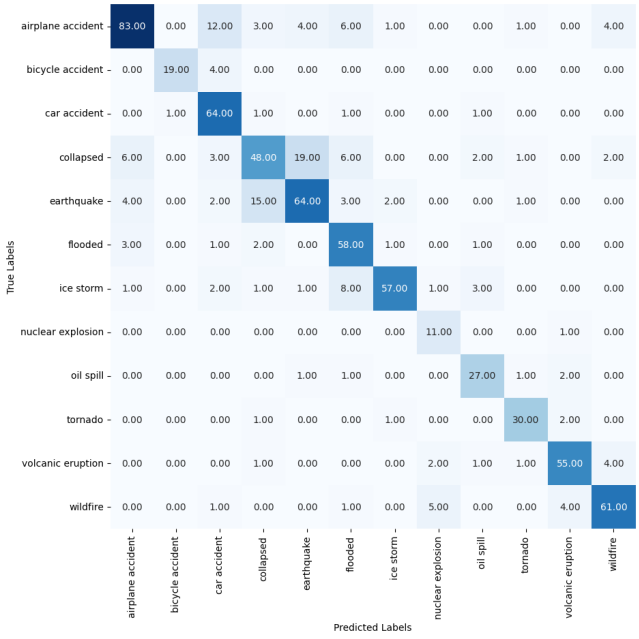

Figure 23: Confusion matrix for vgg

A.4 t-SNE graphs for different transfer learning models

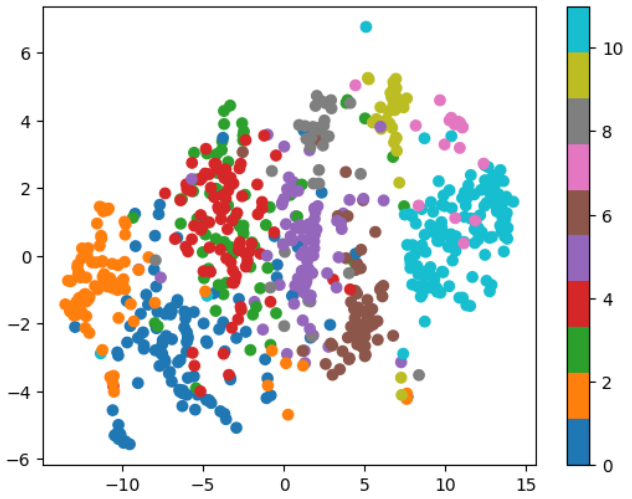

Figure 24: t-SNE graph for Resnet

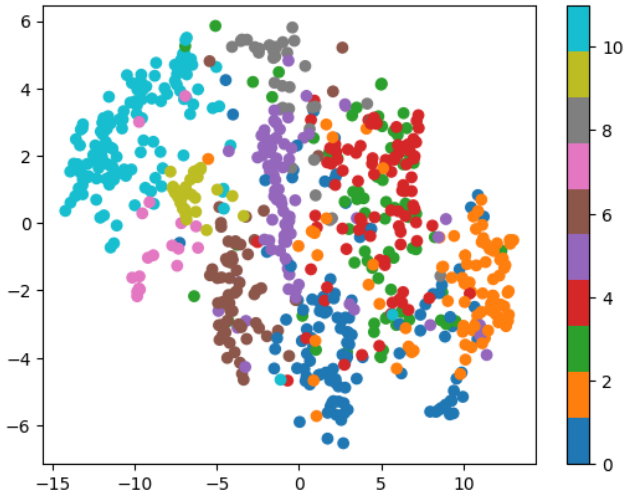

Figure 25: t-SNE graph for alexnet

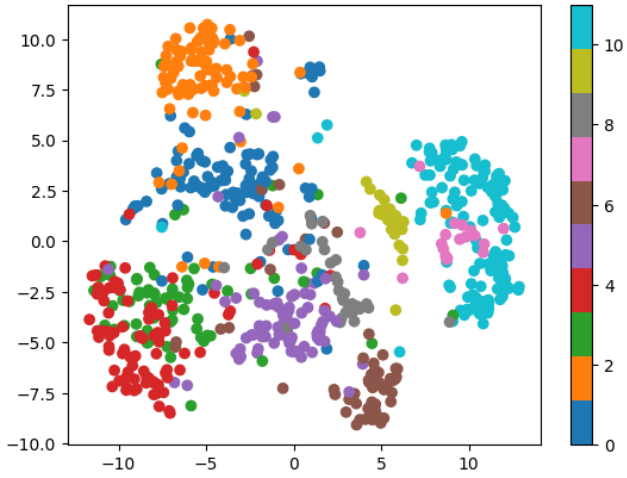

Figure 26: t-SNE graph for densenet

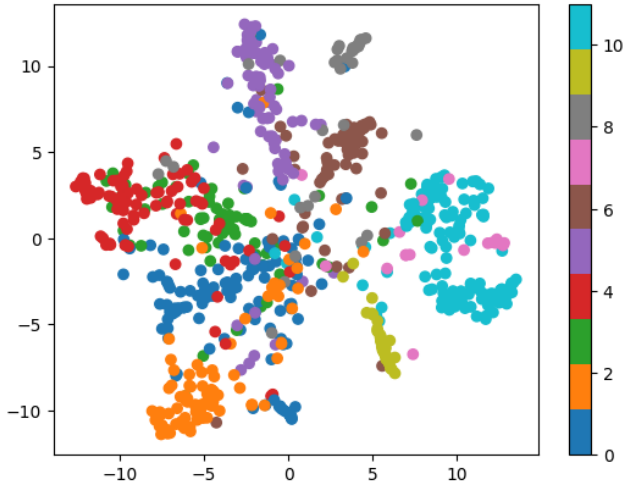

Figure 27: t-SNE graph for squeezenet

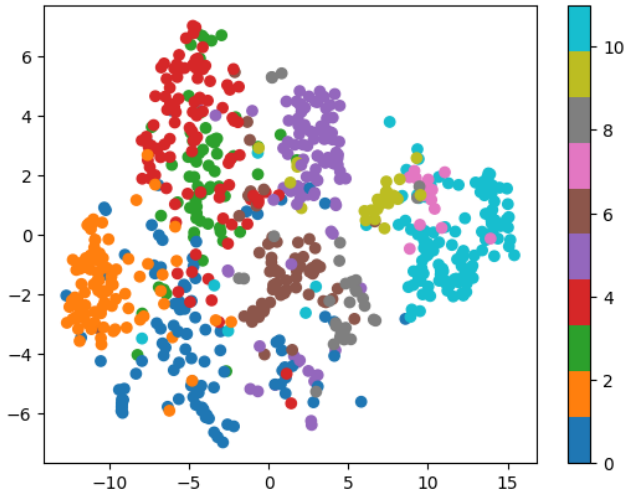

Figure 28: t-SNE graph for VGG