E - CNNs for Material Recognition

CX4042 Neural Networks and Deep Learning
Nanyang Technological University 2022

Filip Rydin: N2202259J@e.ntu.edu.sg
Maëlys Boudier: N2202193F@e.ntu.edu.sg
Maxime Capelle: N2202480A@e.ntu.edu.sg

Table of Contents

Introduction

Object classification from images has historically been widely and deeply explored in research. The same cannot be said about material recognition, which is an inherently difficult task (Muselet et. al, 2019). However, there are numerous practical applications that would benefit greatly from such recognition systems. In manufacturing, deficiencies and impurities could be identified automatically, in recycling, automated systems could be used to distinguish and separate waste and in computer graphics, material recognition could be a building block of algorithms that generate textures.

One dataset that captures the difficulty of material recognition is the Flickr Material Database, FMD (Adelson et. al, 2014). The dataset consists of 10 categories of materials, with 100 images each. The images show a wide range of objects, with different colors and lighting. Half of the images are close-ups whereas the other half are complete pictures of the objects. The large differences between images of the same class make classification difficult, since simple decision rules and features cannot be used.

In this study, Convolutional Neural Networks (CNN) were used on the FMD to try to reach or come close to state-of-the-art performances of other classifiers. Existing architectures were modified and adapted to the specific task using transfer learning. The different hyperparameters in the models were modified to optimize performance. Lastly, the best trained model was applied to a dataset of images of different recycled materials, to test how well the model generalized and explore useful applications.

Related Work

The highest accuracy on the FMD found in literature is reported by Liu et. al. in 2016. The authors use object classification in conjunction with material classification based on the idea that certain objects are made of specific materials. In this way, they achieved an accuracy of 84% on the FMD by first pre-training a model on an extended version of the dataset and then using transfer learning (Liu et. al, 2016). This accuracy can be compared to the human performance of 84.9% on the dataset (Adelson et. al, 2014).

Part of the difficulty of material recognition can be attributed to the difficulty of finding a robust set of engineered features to distinguish different materials. This means that using Convolutional Neural Networks is motivated, since they only require images as input. Other than Liu et. al, several recent papers report high accuracies on the FMD using Transfer Learning with CNNs. In 2015, Bala et. al. achieved an accuracy of 66.5% using a pretrained AlexNet. By combining this approach with inclusion of other features and an SVM-classifier, they were able to improve the accuracy to 69.6% (Bala et. al, 2015). Similarly, Lensch and Wieschollek used Transfer Learning with the LeNet architecture to achieve an accuracy of 64% (Lensch & Wielschollek, 2016).

In a paper from 2013, Sharan, Liu, Rosenholtz, and Adelson outline a model that stands out in terms of performance among the models using engineered features. They were able to reach an accuracy of 57.1% on the FMD using an SVM-classifier and masks to distinguish background from object of interest. Without masks, the reached accuracy was 55.6%. The authors found that color alongside features pertaining to edges in the material work well. They also suggest that global features might be more important than local features to accurately classify materials, since even humans would have trouble distinguishing materials based on close-up images (Adelson et. al, 2013). This reasoning once again suggests that CNNs could be a good approach to solve the material classification problem, since they can be trained to classify images based on both details and global features.

Experiments

In this study, a variety of models were created, and the results were compared and analyzed to determine the optimal configuration. As there are many architectural possibilities but limited computational time and power, the study was commenced by training and testing simpler models to obtain results faster and exclude ineffective configurations. More computational effort was dedicated to the models which demonstrated more promising results. The various steps within the experiments shall be presented in the following section. Codes will be referenced for each of the experiments to be easily reproducible. The specific names of the codes are listed in the Appendix.

Preprocessing Data

Before training the models, the Flickr Material Database had to be preprocessed. Firstly, the data had to be split into three sets of data: train, validation, and test. This was performed using the 'sklearn-train_test_split' function twice in code 1. The distribution of data was as follows: 60% train, 20% validation, 20% test.

Furthermore, it appeared that the dataset contained multiple irrelevant non-jpg files that needed to be removed. It was also discovered that the dataset contained multiple grayscale images which did not have the same depth as the RGB images which were expected. Color is considered an important characteristic of a material and thus it is required that all images in the dataset are in this format. Code 2 performs both these functions and outputs a clean dataset.

Lastly, before the images were fed into the model for training, resizing, and rescaling was performed to obtain the desired image sizes (150, 150) and to normalize the values in the images from values between 0 and 255, to values between 0 and 1. The normalization was performed so that the pixels in the images have similar value distributions allowing for faster convergence (Stöttner, 2019). This operation is performed in all the codes when loading in the images before training the model.

Model Evaluation

To evaluate the quality of the model, a consistent method for evaluation must be determined. In this study, the evaluation was performed with the Three-Way Data Split Method. As previously mentioned, the dataset was split into train, validation, and test sets. The model weights were learned with the train set. The validation set was then used to evaluate the model performance. In this way, multiple different architectures and configurations were evaluated. The best performing model, as determined by comparing the sparse validation accuracy, was then retrained using both the training and validation sets combined. Finally, the test data set was used to evaluate the performance of the retrained model.

To ensure that the best validation accuracy weights were used, a callback of early stopping was implemented which restored the best weights with a patience of 7. This patience value could be considered high for training of 50 epochs, but it was chosen to reduce the chance of the model stopping in minor local minimums. To reduce overfitting during training multiple dropout layers were implemented. Each model's architecture will be detailed in the sections below.

CNN - Vanilla Model

The first model tested was a basic CNN model which provided a baseline score for the classification task. The original comparison baseline was a simple random guessing algorithm that would guess a random class between 0 and 9 with a uniform distribution and achieve the expected 10% test accuracy (alternatively, the algorithm could continuously guess the same class and aim to achieve 10% accuracy). The basic CNN model would provide an improved baseline which would be used to determine if the more complex architectures are effective. The basic CNN architecture consisted of three sets of a convolutional layer, max pooling layer and a dropout layer. After this a flattening layer and dense layer were implemented. The output layer was a 10-unit SoftMax layer for the classification of the 10 classes. The training of this model can be found in code 4. The architecture structure is displayed in Figure 1.

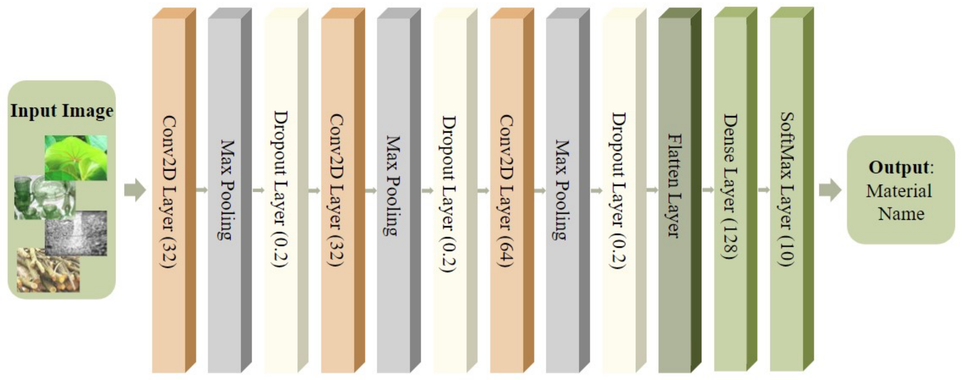

Figure 1: Basic CNN architecture.

This configuration consisted of around 2.6 million trainable parameters and was trained on the regular "FMD_tvt" dataset, which is the data split with the Three-Way Splits method. Once this model was trained, its result was used as a comparison for the more complex models later.

Transfer Learning

After building the vanilla CNN model, the potential of transfer learning models was studied to further increase accuracy. Transfer learning uses pre-trained models to reduce training time and lower generalization error (Mwiti, 2022). The weights of the pre-trained models were frozen, after which a global max pooling layer was added, followed by 3 dense layers and dropouts. The transfer learning models tested were:

- **VGG16**: contains the following 21 layers (16 with weights): 13 convolutional layers, 5 Max Pooling layers, and 3 Dense layers (Rohini, 2021)
- **ResNet50**: consists of a 50 layers deep residual network which implements skip connections in its building blocks (Boesch, 2022)
- **Xception**: consists of a 71 layers deep CNN model - an extension of the Inception structure which uses depth wise separable convolutions (MathWorks, 2022) (Rosebrock, 2017)
- **MobileNet**: consists of 28 layers and has limited parameters (Wang, 2020)
- **DenseNet:** handles the vanishing gradient problem well (Singhal, 2020)
    - **DenseNet121**: 121 layers deep
    - **DenseNet201**: 201 layers deep
- **EfficientNetV2B0**: this model was built to "reduce parameter size and FLOPS" (Tan et. al, 2019)
- **InceptionV3**: consists of a 48-layer model which serves as a "multi-level feature extractor" (Rosebrock, 2017)

The layers that were added after the transfer learning model can be seen below in Figure 2.

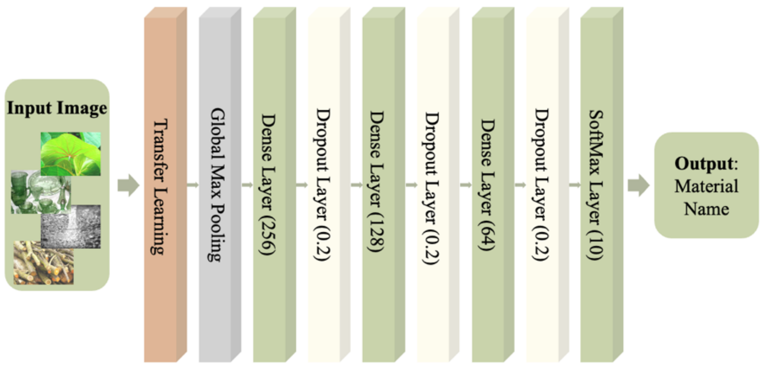

Figure 2: Transfer Learning Model Architecture

For the training of the transfer learning models, code 5 can be applied and run with the “FMD_tvt” dataset. Code 5 acts as a template for the transfer learning models as one can simply change the pre-trained model that is imported into the code by changing one line of code when building the base model. The limitations of this template are that the pre-trained architectures available are limited to the ones found in “tensorflow.keras.applications”.

Data Augmentation

To achieve a high level of accuracy on unseen data, it is crucial to have a large dataset that exposes the network to a variety of images of the same class. This is required as it allows the weights to be learned in such a way that it recognizes the general patterns that define a given class, rather than the specific features in the training data samples. Due to there being limited data provided for the given classes in the FMD dataset, other methods such as data augmentation were applied to introduce increased variety within the data samples.

In this study, the python library “Albumentations” was used to simultaneously augment the provided image and its corresponding mask. With this library a large variety of changes can be applied. For the FMD dataset, the following augmentations were deemed relevant and beneficial for improved training of the material types:

- Random crop (width = 256, height = 256)
- Horizontal and Vertical flip (p = 0.5)
- RGB shift (p = 0.4)
- Color Jitter (p = 0.2)
- Random Brightness Contrast (p = 0.2)

For the initial training of the constructed models, the original FMD dataset of 200 images and masks for each of the 10 classes was used to keep a lower training time. Once the optimal models had been established, the augmented FMD dataset was used to further fine tune the models and achieve improved accuracy. The data augmentation is performed in code 3 and creates a new folder called ‘FMD_aug_tvt’. In the code, the data augmentation is applied on both the train and validation set. The train set images were each transformed 10 times, whereas the validation set images were only transformed five times each. These hyperparameters were determined by choosing a value which was believed to sufficiently enhance the data without significantly increasing the computational time. If the previously mentioned models, trained in code 4 & 5, need to be retrained with the newly created augmented data, ‘image_folder_name” must be changed in the inputs to the newly created ‘FMD_aug_tvt’. An example of such a transformation is given in Figure 3.

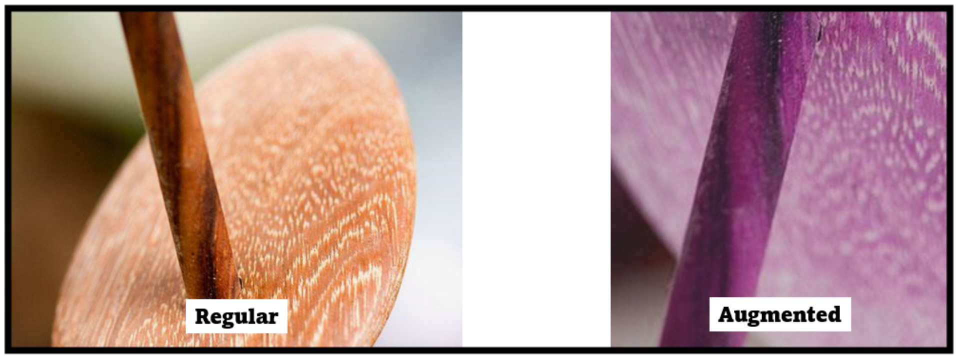

Figure 3: Example comparison of regular and augmented images.

Keras Tuner

Hyperparameter tuning is important as it may help improve the accuracy of the model. Model-based hyperparameters tuning was implemented to determine the optimal number of neurons in each of the three dense layers, the learning rate, and the dropout probability. The *keras-tuner* library was used to pick the optimal hyperparameters and only run on some of the highest achieving baseline models as the tuning algorithm is computationally demanding (Singh, 2021). The goal is to find the optimal configuration of hyperparameters to train the model and get the highest accuracy (sparse validation accuracy was used for comparison purposes) by trying many variations of the baseline model. The keras tuner was applied in code 6 with the ranges shown in Table 1:

Table 1: Keras tuner hyper parameter ranges and steps

Hyper Parameter	Ranges	Step Size
Units Dense layer 1	(128 – 512)	128
Units Dense layer 2	(64 – 256)	64
Units Dense layer 3	(16 – 64)	16
Learning rate	(0.0001 – 0.01)	Log sampling
Dropout probability	(0.1 – 0.5)	0.1

Using the Masks

Masking images involves filtering out certain pixels. The Flickr Material Database contains both images and masks featuring only the main object to analyze. This means that the important part of the images can be extracted and used to train models instead of the whole images. The goal is to train the model to focus on what is important and filter out elements that could interfere with the classification.

Generating such masks can be difficult however and might not be feasible in all applications. As such, masking was used only on the training data. The trained model was then evaluated on non-masked validation data to see whether accuracy had improved compared to not using masks for training. This was done as a last step after data augmentation to determine the optimal model. The application of the masks to the augmented data set is done through code 7, while the training of a model on the masked data set can be done by modifying code 5.

Results

In this section, the results from the experiments will be explained and compared to determine which model is the best for the material classification task. The best model was also run on a new dataset - Kaggle's Garbage Classification dataset - to test its generalization ability.

Basic CNN Results

Using the vanilla CNN model, an accuracy of 23.5% was achieved. This is a significant improvement from the baseline of 10% but is still far from human accuracy and accuracy reported in literature. As such, there is room for improvement.

Determining the Best Model

The validation accuracies in Table 2 were achieved with the basic Transfer Learning models and the previously outlined final layers (see Figure 2). Xception and DenseNet201 performed best and were tuned to optimize hyperparameters. Notably, some of the models performed worse than the baseline CNN model and were unable to converge (e.g., ResNet50).

Table 2: Validation accuracies for different Transfer Learning models.
(Best performing marked in bold)

Model	VGG16	ResNet50	**Xception**	MobileNet
Val Accuracy	0.5728	0.1507	0.7739	0.7487
Model	InceptionV3	DenseNet121	**DenseNet201**	EfficientNetV2B0
Val Accuracy	0.6784	0.7538	0.7839	0.1608

Tuning Results

The final validation accuracies after tuning are displayed in Table 3, along with the optimal hyperparameters. Note that the validation accuracy for DenseNet201 has improved, while the supposed optimal parameters for the Xception model do not seem to be better than the original. As such, Xception with the original parameters and DenseNet201 with the tuned parameters were used to conduct further tests using augmented data.

Table 3: Validation accuracies for different Transfer Learning models after tuning.

Model	**Val Accuracy**	**Learning rate**	**Dropout rate**	**Size dense layers**
DenseNet201	0.7940	0.0023	0.4	256/192/48
Xception	0.7549	0.00017	0.3	256/128/48

Augmentation and Masking Results

The validation accuracies after training on augmented data are displayed in Table 4, along with the validation accuracy after training the best model on augmented masked data. The Xception model trained on the data without masking performed better on the validation set and has the best validation accuracy of the models studied in this project. As such, this model is trained on both training and validation data and tested on test data.

Table 4: Validation accuracies for models after training on augmented and masked data

Model	**Data**	**Val Accuracy**
Xception	Augmented	0.8442
DenseNet201	Augmented	0.6884
Xception	Augmented & Masked	0.7328

Final performance on test data

Now that the Xception Model with Augmented Data had been determined as the best architecture, the model was trained again with the train and validation set combined into one and was tested with the previously untouched test data set. Retraining of the best model is performed in code 10. This test dataset gives the best estimation of the true error however, it will be an optimistic value. The results are displayed in Table 5 below:

Table 5: Results from final training of best model

Model	Data	Test Accuracy
Xception	Augmented - (Train + Validation)	0.735

For the training, the model was rapidly becoming overfit. As such, the training was terminated after only 8 epochs due to the patience value of p = 7. The train and test accuracies on different epochs are shown in Figure 4 below. Overfitting rapidly occurs as the training accuracy continuously increases whilst the test accuracy fluctuates around the same value of 0.70. There is the same relationship between the train and test losses which again suggests that overfitting is occurring.

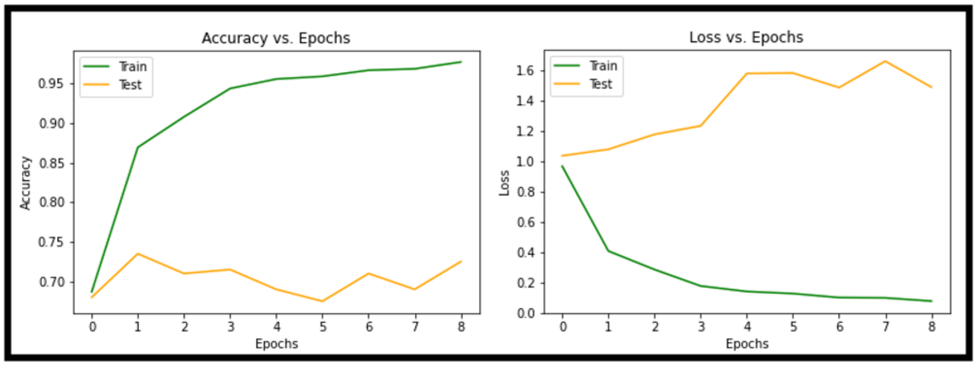

Figure 4: The accuracy and loss plots vs epochs for the final model.

To further analyze how effectively the materials were classified, a confusion matrix was constructed, and the relationships were calculated. By finding the true positives, true negatives, false positives and false negatives, multiple other comparison metrics could be calculated. Precision was deemed the most representative parameter in this case and was found using the equations described in "*Artificial Intelligence-Based Brain-Computer Interface*" (Demir, 2022).

Table 6: Precision values for each of the values

Class	Precision
Fabric	0.667
Foliage	**0.857**
Glass	0.714
Leather	0.824
Metal	0.654
Paper	0.8
Plastic	**0.615**
Stone	0.812
Water	0.783
Wood	0.714

Table 6 demonstrates the final precision scores for each class. The model performed best on the Foliage class, whereas it performed worst on the plastic class. This result aligns with Adelson's paper which claims that materials such as metal and plastic have surfaces that include less specificities (since they can be molded into many shapes) than materials like foliage (natural objects with less variation) and thus make it harder to classify (Adelson et. al, 2013).

Running Best Model on Garbage Classification Data

All the experiments performed so far were based on the FMD dataset which utilizes the same types of images from similar perspectives. Therefore, to see the actual utility of the model, the final Xception model was applied to the Kaggle Garbage classification dataset. This dataset contains 6 types of classes:

cardboard, glass, metal, paper, plastic, and trash. The images in the Garbage classification dataset give a more accurate representation of 'real life' application as the images consist of isolated pieces of garbage of different materials. The FMD has 4 classes in common with this dataset and the model was only tested with those four classes. The classes tested were glass, metal, paper, and plastic. In Figure 5, images of the two datasets are compared.

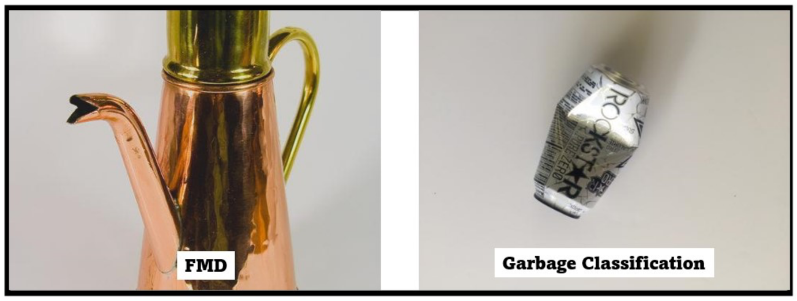

Figure 5: Comparison of dataset images (class metal)

Code 10 uses the best model obtained from code 9 ('Final_Training_Xception_tt.h5') and applies it on the garbage dataset to obtain predictions. The result is shown in Table 7 below. As shown, a classification accuracy of 52.1% was achieved, which means that the model performs significantly better than simply guessing. This demonstrates that the model was able to sufficiently generalize to be applied to different types of datasets. If this result needed to be further improved, one would tailor the training data more to the garbage data.

Table 7: Results from running model on Garbage classification data.

Model	Data	Classification Accuracy
Final_Training_Xception_tt	Garbage classification data	0.521

Discussion

In this section, the obtained results, and their comparison to models in literature will be analyzed. Next steps, particularly regarding improving accuracy on the garbage classification dataset will also be discussed.

Discussion of results

One surprising result was that masking the training data did not improve accuracy on non-masked validation data. The goal was to teach the model to focus on important parts of the images. However, when doing so, the training data becomes different from validation data, which seems to affect the performance more than the positive effects. Other approaches to using the masks could be tried to potentially achieve better results. One such approach could be to only mask a smaller part of training data. It is also likely that performance would improve if the masks were applied to validation and test data as well, but to generalize the results the masks were determined redundant as they limited applications (e.g., the garbage dataset provides a full image with a background and no masks).

Another surprising result was that the tuner did not have a significant effect in increasing validation accuracy. One reason could be that only ten sets of parameters were tested with Keras Tuner, while more would have been needed to find the true set of optimal parameters. More parameters could also have been optimized, such as the number of hidden layers added to the end of the base networks. Additionally, ImageNet weights were always used for the transfer learning models. It is possible that using the specific model weights could have yielded better results.

Comparison with Performance in Literature

The final accuracy of 0.735 is higher than that for many of the models used in literature. However, it is not as high as state-of-the-art performance or human accuracy. At the same time, the workflow used by Liu et. al. to develop the state-of-the-art model is highly specific to solve the task of material classification, whereas the workflow employed in this study is more general. It should also be mentioned that the authors of the article pretrain on another material recognition dataset. This means that our approach, although it produces a less accurate model, might be more accessible to use by professionals in practise, which makes it highly suitable for applications in industry. As such, the main contribution of this study is the general workflow applied to find a suitable model for good-enough material recognition. The results on the garbage classification data show that a model developed in this way using the FMD can generalize to new data.

Next Steps

Although our model only classified 52% of the garbage dataset correctly, there is an opportunity to re-train the model with a mix of the Flickr and garbage data as the images in both datasets are slightly different. Furthermore, the garbage dataset had two additional categories which could also be beneficial to a material detection algorithm. In Europe, *Recycleye Vision* has already implemented a computer vision-based software to sort waste (Willis, 2022). As companies are evolving to include sustainability as part of their operations, building a good waste sorting mechanism is crucial and computer vision gives us the tools to do so. Environmental benefits of recycling include “conserving energy, reducing air and water pollution, reducing greenhouse gases, and conserving natural resources” (Stanford, 2021). Overall, endeavors to accurately classify materials are beneficial to help reduce waste in landfills and promote proper recycling of materials.

References

Adelson, E. H., Liu, C., Rosenholtz, R., & Sharan, L. (2013). *Recognizing Materials Using Perceptually Inspired Features.* International Journal of Computer Vision, 108(3), pp. 348-371. https://doi.org/10.1007/s11263-013-0609-0

Adelson, E. H., Rosenholtz, R., & Sharan, L. (2014). *Accuracy and speed of material categorization in real-world images.* Journal of Vision, 14(9), article 12. https://doi.org/10.1167/14.9.12

Bala, K., Bell, S., Snavely, N., & Upchurch, P. (2015). *Material Recognition in the Wild with the Materials in Context Database.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.48550/arXiv.1412.0623

Boesch, G. (2022). *Deep Residual Networks (ResNet, ResNet50) 2022 Guide - viso.ai*. Viso Suite. Retrieved November 3, 2022, from https://viso.ai/deep-learning/resnet-residual-neural-network/

Demir, F. (2022). *Artificial Intelligence-Based Brain-Computer Interface* Retrieved November 11, 2022, from https://www-sciencedirect-com.tudelft.idm.oclc.org/book/9780323911979/artificial-intelligence-based-brain-computer-interface

G, R. (2021, September 23). *Everything you need to know about VGG16 | by Great Learning*. Medium. Retrieved November 3, 2022, from https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

Lensch, H. P. A., & Wieschollek, P. (2016). *Transfer Learning for Material Classification using Convolutional Networks.* ArXiv. https://doi.org/10.48550/arXiv.1609.06188

Liu, X., Okatani, T., Ozay, M., & Zhang, Y. (2016). Integrating deep features for material recognition. 23rd International Conference on Pattern Recognition (ICPR). https://doi.org/10.1109/ICPR.2016.7900209

MathWorks. (2022). *Xception convolutional neural network - MATLAB xception*. MathWorks. Retrieved November 7, 2022, from https://www.mathworks.com/help/deeplearning/ref/xception.html

Muselet, D., Tremenau, A., & Xu, S. (2019). *Deep Learning for Material recognition: most recent advances and open challenges.* International Conference on Big Data (BIGDML). https://doi.org/10.48550/arXiv.2012.07495

Mwiti, D. (2022, July 21). *Transfer Learning Guide: A Practical Tutorial with Examples for Images and Text in Keras - neptune.ai*. Neptune.ai. Retrieved November 3, 2022, from https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

Rosebrock, A. (2017, March 20). ImageNet: VGGNet, ResNet, Inception, and Xception with Keras. PyImageSearch. Retrieved November 7, 2022, from https://pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

Singh, A. (2021, August 5). Hyperparameter Tuning Of Neural Networks using Keras Tuner. Analytics Vidhya. Retrieved November 11, 2022, from https://www.analyticsvidhya.com/blog/2021/08/hyperparameter-tuning-of-neural-networks-using-keras-tuner/

Singhal, G. (2020, May 6). *Introduction to DenseNet with TensorFlow*. Pluralsight. Retrieved November 7, 2022, from https://www.pluralsight.com/guides/introduction-to-densenet-with-tensorflow

Stanford. (2021). Frequently Asked Questions: Benefits of Recycling | Land, Buildings & Real Estate. Land, Buildings & Real Estate. Retrieved November 11, 2022, from https://lbre.stanford.edu/pssistanford-recycling/frequently-asked-questions/frequently-asked-questions-benefits-recycling

Stöttner, T. (2019, May 16). Why data should be normalized before training a neural network. Towards Data Science. Retrieved November 6, 2022, from https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d

Tan, M., & Q. V. (2019, May 29). *EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling*. Google AI Blog. Retrieved November 7, 2022, from https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

Wang, D. (2020). *A Novel Image Classification Approach via Dense-MobileNet Models*. Hindawi. Retrieved November 7, 2022, from https://www.hindawi.com/journals/misy/2020/7602384/

Willis, S. (2022, June 24). How Computer Vision Has Evolved and Its Role in Waste Sorting. Recycleye. Retrieved November 11, 2022, from https://recycleye.com/how-computer-vision-has-evolved-and-its-role-in-waste-sorting/

Appendix

CZ4042-Codes Provided:

- Code 1: “1. Split_FMD_TrainValTest”
- Code 2: “2. Remove_Wrong_Format_Files (FMD_tvt)”
- Code 3: “3. Data_Augmentation”
- Code 4: “4. Basic_CNN”
- Code 5: “5. Transfer_Learning_Template”
- Code 6: “6. Keras_Tuner_Transfer Learning”
- Code 7: “7. Apply_mask”
- Code 8: “8. Split_TrainTest”
- Code 9: “9. Final_Transfer_Learning_Code”
- Code 10: “10. Garbage_Classification_Tester”
- Code 11: “11. Performance_Calculator”

Data Provided

- FMD dataset
- Modified Garbage Classification set (removed non-overlapping classes)