COMP5329 Deep Learning - Semester 1 2021: Assignment 2

Group members:
Jesse Serina Narvasa(500525438),
Patrick McLennan (500448953),
Sushmita Shrikrishna Jadhav(500393860)

May 30, 2021

1 Introduction

1.1 Aim of study

The report aims to perform multi-label classification on a given dataset by implementing various convolutional neural networks (CNN) including state of the art architectures known within the deep learning community. The report focuses on a comparative analysis between the different Deep ConvNets' performance, along with analysis of its components that allows it to achieve such performance for this particular problem. Thereafter, potential modifications are then considered that may further improve performance based on available information within the dataset, as well as modifying components within the network to take advantage of its effects. Through this, a holistic view of the best CNN architecture that suites this dataset and problem can be determined, along with the deduction of how each component contributes to the overall effectiveness of the model.

1.2 Importance of study

In many real-life deep learning applications, there is a set of suitable tags which can be assigned to any given image, and not just a single tag. Scikit-learn states that multi-label classification works by assigning a set to target labels to each sample (scikit learn, 2021), whereas multi-class classification assumes that only one label from the set of labels can be assigned to each sample (Bebnev, 2021). To perform operations like object detection it is essential to successfully perform multi-label classification as there can be many identifiable objects in a single view or single image. For example the below image, contains identifiable elements like person, car, tree, snow and skis in one single image. The caption of the given image states: A "person standing in the snow next to a car on skis".

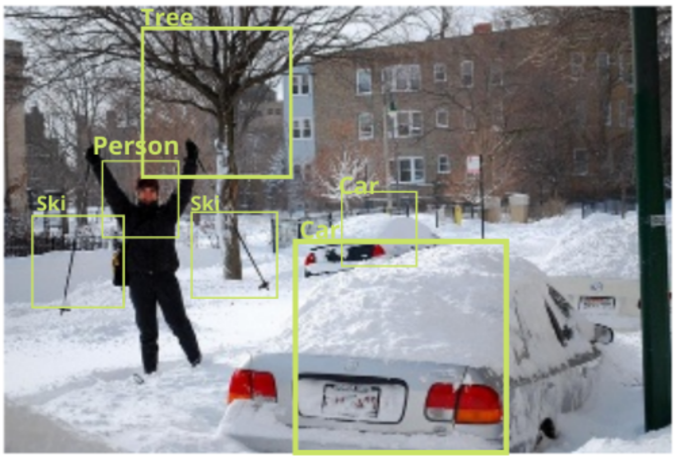
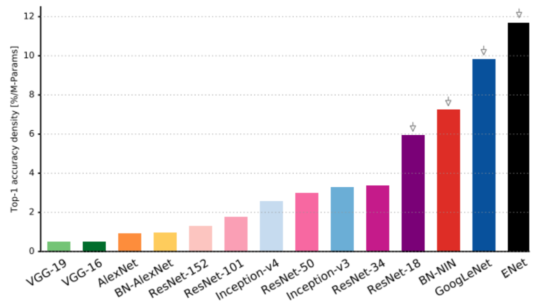


Figure 1: Multi-label classification Example

There are various image datasets available to build a deep neural network architecture like ImageNet dataset and multi-label datasets like MS COCO, Open Images which have more objects per image (Durand, Mehrasa, & Mori, 2019). These dataset are usually generated through crowd-sourced data, in which noise can take place in the form of inaccurate labels. These datasets also include captions which provides a descritpion of the image. The inclusion of captions within the dataset therefore provides the opportunity to for increased amount of information available for the model during information and prediction. The effective use of this additional information should improve prediction performance.

1.3 Previous Work

To achieve a better label predictions underlying neural network architecture plays a key role. The neural network model needs to perform effective feature extraction to improve accuracy whilst managing the computation efficiency. Many architectures like AlexNet, Convolutional Neural Network, ResNet, GoogleNet, DenseNet, etc have been implemented successfully to perform multi-class classification. The accuracy of various State-of-the-art deep neural networks to the ImageNet dataset were noticed, which showed that Enet is the best architecture design followed by GoogleNet (Canziani, Paszke, & Culurciello, 2017). The



Accuracy per parameter *vs.* network. Information density (accuracy per parameters)

Figure 2: Accuracy per parameter vs Network for ImageNet (Canziani et al., 2017)

best Deep Neural Architecture model selection the model analysis was based on per consumption (independent of batch size), model accuracy and inference time, model complexity, and number of parameters and operations which effected the inference time. There are other architectures which have used visual attention method to selectively process the most important visual data input to help faster and accurate image data understanding (Luo, Jiang, & Zhao, 2019).

1.4 Dataset

Dataset plays a very sensitive role in deep learning algorithms. The dataset provided for this multi-label classification contains 40,000 images of different sizes, and two .csv files used to split the dataset training and testing sets. There are total of 19 classes or labels which are tagged to the image samples in the training dataset. Along with the labels the training dataset samples have captions which define the image specifically the action or activity performed in the image sample.

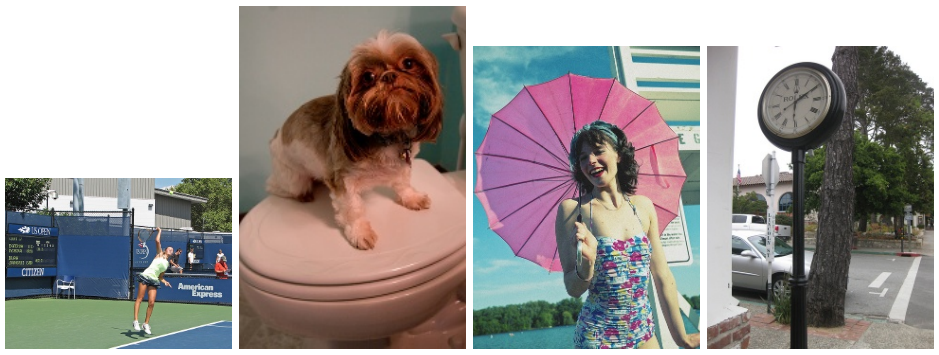


Figure 3: Sample Images from Dataset

The main drawback of the dataset is label imbalance. Label imbalance is a generally found issue in various datasets, where the dominance of a one or a small subset of label values overshadows the relevance and predication of other labels. For the given dataset, label 1 is dominant in the dataset. A bubble chart was created to analyse the dominant variables which helped further in weight initialization.

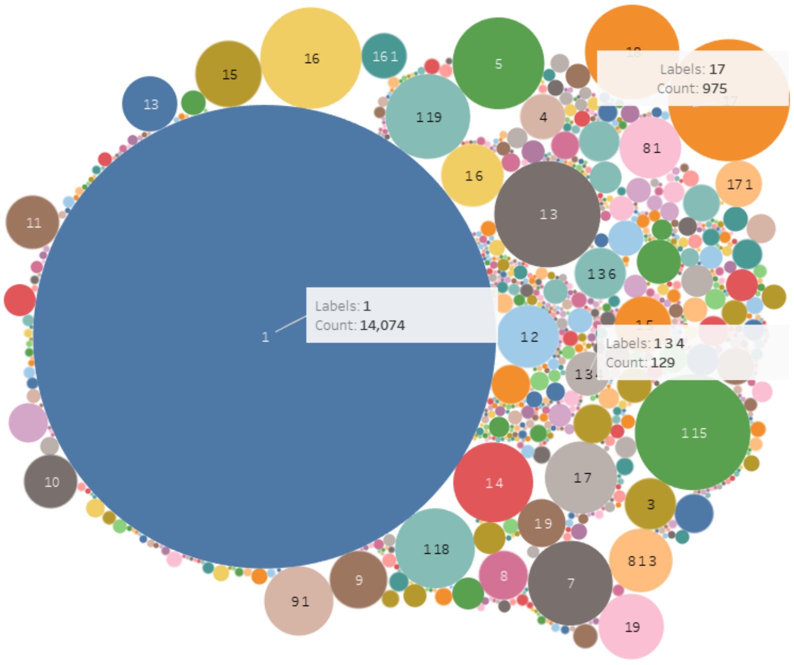


Figure 4: Dominance of label 1 in the dataset

2 Methods

2.1 Data Preprocessing

The dataset was pre-processed by performing various transformations to ensure data used to training model are in desired form. The transformations used include:

2.1.1 Normalisation

Min-Max Normalisation Min-Max Normalization is performed on the images to perform feature scaling. Images have a pixel density from 0-255 which are scaled down by using normalization techniques to scale of 0-1 (Varsheni, 2021). Whilst this is not critical, considering that the minimum and maximum values are constrained to 0 and 255 respectively, this min-max normaisation is applied by default when converting the image to a Torch tensor within the ToTensor() function.

Standardization Another prominent feature scaling technique is Standardization which centers the data points around mean pixel value with a standard deviation value. In order to conduct this transformation, the mean and std dev. values were first calculated for the dataset, thereafter, the standardisation is then applied with the calculated values as inputs. Also known as whitening, this transformation ensures that the data is centred within the dataset's mean value per channel, along with the spread stated within the std. deviation, also applied per channel.

2.1.2 Data Augmentation Techniques

Random Horizontal Flip This Data Augmentation technique used to flip an image sample on its horizontal axis. The image samples in the dataset were randomly flipped based on a probability factor which was defaulted to 0.5 . Through this, for each epoch, the data will therefore have the probability of 50% of it being flipped, therefore each epoch would provide the model with slight variations within the dataset, and hence achieve augmentation.

Random Vertical Flip Similar to Random Horizontal Flip above, randomised vertical flip has been added as means of augmentation for our model training to allow for increased randomness in the data being provided to the model, and hence increase its robustness.

Converting images to RGB mode A few images in the dataset were observed to be in black and white color. To ensure consistency, all the images were converted in to RGB mode(as coloured images were dominant).

Functional Transformations Additional fine-grained feature scaling transformations were applied to the image samples to enhance brightness, contrast, hue, saturation and sharpness.

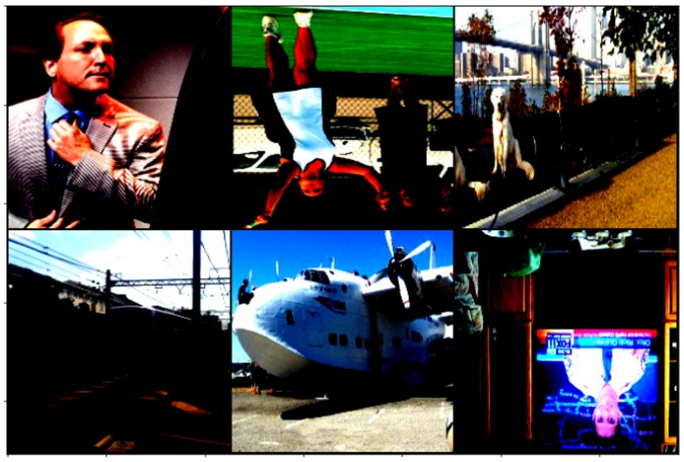


Figure 5: Images after transformations applied

2.2 Solving label Imbalance

Weighting Approaches
The data was found to contain many instances of '1' as label to an extent where it constitutes over a third of the dataset. To overcome this, random sampling by weights was attempted by using the Pytorch WeightedRandomSampler to balance the data by applying the inverse of the occurrence of a label to the sampler, but given the nature of the imbalance it was found the balancing labels with very low occurrence was inefficient as this just resulted in single observations appearing multiple times in the adjusted dataset. An alternative approach to this was to utilise the BCEWithLogitsLoss and setting pos weights in a similar manner to the WeightedRandomSampler approach, however it was ultimately found that incorporating MultiLabelSotMargin loss in training resulted in stronger performance regardless of the label imbalance. Given F1 scores were still strong when many '1' classes were predicted, this implies that the '1' class is of high occurrence in the test data hence the strong performance without label balancing.

2.3 Loss Functions

BCEWithLogitsLoss is used for Binary Cross Entropy Loss which combines a BCEloss and Signmoid layer in one single class. This combination in one single lat=yer makes the loss function is numerically stable we take advantage of the log-sum-exp trick for numerical stability (BCEWithLogisticsLoss, 2021). The formula is as follows:

$$BCEWithLogitsLoss(x, y) = L_{n,c} = -w_{n,c}[p_c y_{n,c}.log\sigma(x_n) + (1 - y_{n,c}).log\sigma(x_{n,c}))] \tag{1}$$

where c is the label number, n is the number of the samples in the batch and p_c is the weight for the class c (BCEWithLogisticsLoss, 2021).

MultiLabelSoftMarginloss The MultiLabelSoftMarginloss optimizes a multi-label one-vs-all loss based on max-entropy, between input x and target y of size (N, C)(MultilabelSoftMarginLoss, 2021). So, loss for every sample in the mini-batch is as follows :

$$MultiLabelSoftMarginloss(x, y) = -\frac{1}{C} * \sum_i y[i] * log((1 + exp(-x[1]))^{-1}) + (1 - y[i]) * log(\frac{exp(-x[i])}{(1 + exp(-x[i]))} \tag{2}$$

In actually both the loss function produce the same output by deadalut, but the main difference is that self-generated weights can be assigned to BCEWithLogitsLoss where as same is not possible with MultiLabelSoftMarginloss.

2.4 Validation metrics

2.5 F1 Score

F1 Score also called as Balanced F-score or F-measure is used in this study for validation. The F1 score has a range between (0,1) where 1 is the best and 0 is the worst score. For F1 score, both precision and recall have equal contribution. The formula is as follows:

$$F1 = 2 * (precision * recall)/(precision + recall) \tag{3}$$

For Muli-label classification, Average F1 score is calculated which is the Average of F1 score of each label.

3 Convolutional Neural Network Design

3.1 CNN Overview

CNN Architecture is one of the prominent architecture which efficiently handles image datasets. It has 3 important components: Convolutional layers, pooling, and fully-connected layer(Nelson, 2021). In

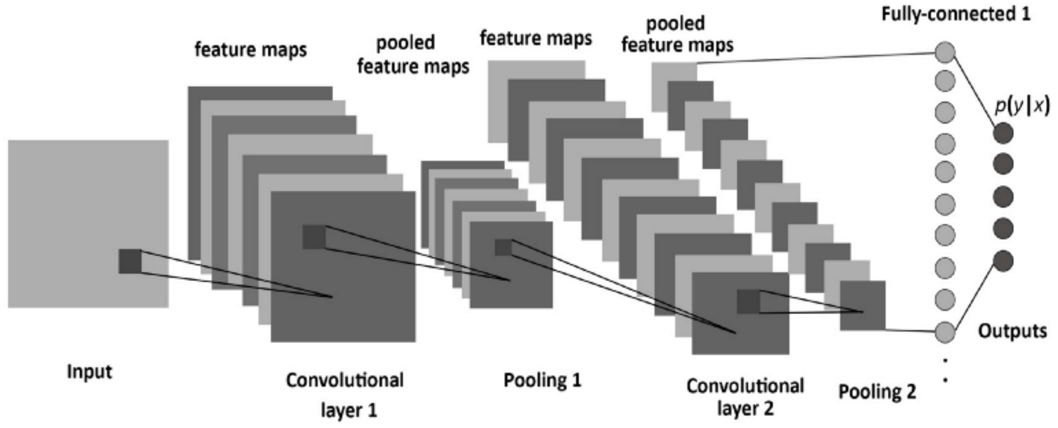


Figure 6: Basic CNN Architecture(Nigam, 2021)

CNN model, every image is convolved over by a slidding the filter (kernel) over the image to perform modification. Pooling is then applies majorly to reduce the spatial size of the convolved image i.e. performs dimension reduction to help with reduce computation power. Pooling also helps in extracting the dominant features which helps in training the model effectively ("Lecture 7: Neural Netwrok Architecture, Lecture Slides, COMP5329: Deep Learning, The University of Sydney, delivered 26 March 2021.", 2021). This way many convolutional layers and pooling layers are formed which are then flattened into a fully connected layer. This fully connected layer further generates output i.e. class or label of the image.

3.2 AlexNet

AlexNet is a modified version of CNN with more deeper structure. It has significant improvement compared to CNN on image classification problems ("Lecture 7: Neural Netwrok Architecture, Lecture Slides, COMP5329: Deep Learning, The University of Sydney, delivered 26 March 2021.", 2021). The main upgrade and advantage of AlexNet was its ability to leverage GPU for training (Brownlee, 2021). This helped the architecture to train large number of parameters and increase in performance.

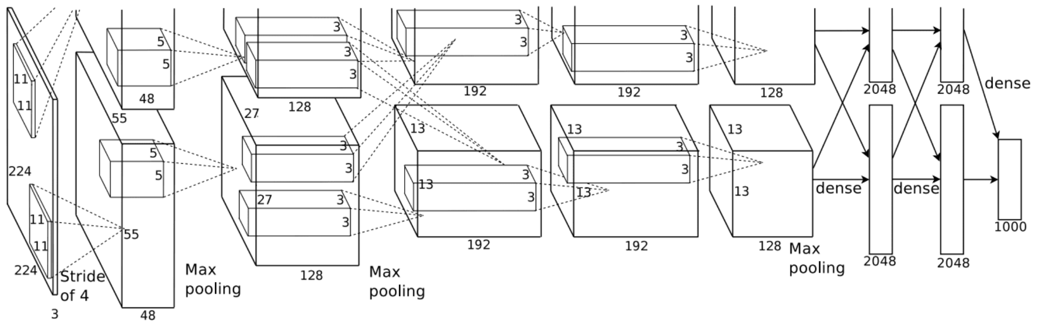


Figure 7: AlexNet Architecture (Brownlee, 2021)

3.2.1 Architecture of AlexNet

AlexNet Architecture is made of eight layers: 5 convolutional layers which include max-pooling layers and 3 fully connected layers ("Lecture 7: Neural Netwrok Architecture, Lecture Slides, COMP5329: Deep Learning, The University of Sydney, delivered 26 March 2021.", 2021). Relu Activation function is applied to each layers excluding the output layer. Relu is used as it increases the training speed by almost a multiple of 6 and also because it helps prevent saturation(Brownlee, 2021). Dropout layers are also applied between fully connected layers to avoid over-fitting of data("Lecture 7: Neural Netwrok Architecture, Lecture Slides, COMP5329: Deep Learning, The University of Sydney, delivered 26 March 2021.", 2021).

3.3 ResNet

ResNet model or Residual Network Architecture was formed as a solution of testing the hypothesis: Deeper modules are harder to optimize. ResNet generates a very deep neural network using residual connection or residual blocks.

3.3.1 Architecture of ResNet

The model contains majorly 3*3 convolutional layers (152 layers) ("Lecture 7: Neural Netwrok Architecture, Lecture Slides, COMP5329: Deep Learning, The University of Sydney, delivered 26 March 2021.", 2021). Average pooling used to remove fully connected layers and also dropout is not used. The main feature of ResNet is the use of residual connection which help the network connect with the original input (non weighted input)("Lecture 7: Neural Netwrok Architecture, Lecture Slides, COMP5329: Deep Learning, The University of Sydney, delivered 26 March 2021.", 2021). This help in better learning as the original image's features are extracted in the deep network.

3.4 Inception Version 3

The Inception architecture introduced a key concept called inception module, which is a block of convolutional layers with filters of different size like 1×1, 3×3, 5×5 with an additional 3×3 max pooling layer. Implementing these 3×3, 5×5 filter is computationally expensive for which inception architecture performs factorization("Lecture 7: Neural Netwrok Architecture, Lecture Slides, COMP5329: Deep Learning, The University of Sydney, delivered 26 March 2021.", 2021). For example, a 3×3 filter is factorized into 3×1 and 1×3 filters. Inception Version 3 architecture also performs Label Smoothing by adding

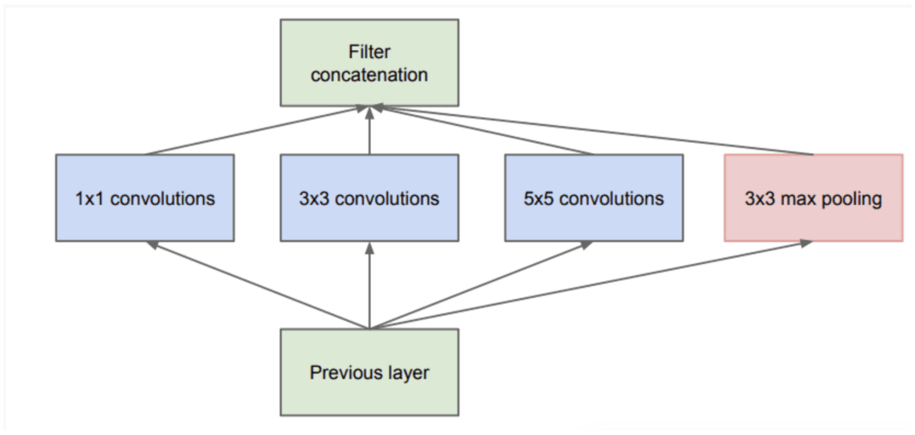


Figure 8: Inception module

noise during label distribution("Lecture 7: Neural Netwrok Architecture, Lecture Slides, COMP5329: Deep Learning, The University of Sydney, delivered 26 March 2021.", 2021).

3.5 Design of Novel Architecture

In addition to using pre-trained models to evaluate its effectiveness against the dataset, modifications to the existing state-of-art models were also considered to see if it would improve its performance.

Amongst these modifications, were the inclusion of caption data being fed into the models above. This has been considered, as the dataset also includes caption data along with the images, and considering that the captions include keywords which may provide additional information to the type of image, then it will be useful to provide the additional information to the neural network.

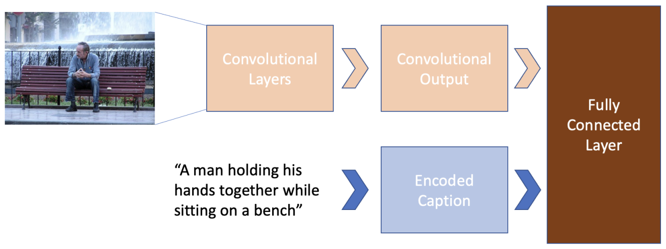


Figure 9: Architecture design with image convolution and caption encoding

In order to append this information to the neural network, it has been decided to provide the additional data to the fully connected layer at the end of the models. Models such as ResNet which do not include a fully connected layer meant that it had to be appended. This results in two sets of data being provided into the fully connected layer, the first being the image output after the convolutions, and the second set of data would be the encoded caption. This means that the fully connected layers has the job of mixing the convolution data as well as the encoded captions and therefore has more information in deciding the labels per image.

3.5.1 Three Letter Caption Encoding

In order to make use of the caption data, the captions are encoded with the string being converted into lower-case and then stripped of non-alphabet characters. We then form three letter combinations which where the frequency of each three-letter combination is stored in an array, which will be the encoded form of the caption.

Whilst this approach may be simple, the main objective of this caption encoding is to capture the additional information stored within the image captions. While storing individual words would be simpler, it does have a large drawback whereby words which it did not encounter during training will be unrecognised during prediction and hence be discarded. As a result, encoding by letter combinations over word encoding was much more favourable.

Additionally, models such as LTSM(Long-Short Term Memory) were considered in order to process the caption data. However, it is important to note that the main advantage of LTSM is in being able to store previous information for sequential data such as sentences. While captions would qualify for sequential data, LTSM would have been more useful for applications such as sentiment analysis, where in this instance, we are simply interested in words which are present within the caption. Specifically, we are interested in the model being able to recognise three letter combination/s being associated with certain labels such as "b-u-s" being associated with class 6. Therefore, whilst LTSM may still be useful, the simplicity and the advantage of not having to have additional parameters to train with just encoding the caption data has been selected.

3.5.2 Custom Models

With the proposal of the novel architecture described above, this approach has been implemented on all of the existing architectures previously mentioned including AlexNet, ResNet, and Inception models. This is later referred to as Custom AlexNet, Custom ResNet, and Custom Inception respectively.

The modification has been performed such that the fully connected layer within AlexNet and Inception also accepts the encoded caption data. This has been implemented by increasing the expected input size for the first fully connected layer. Consequently, since ResNet does not have a fully connected layer, one has been appended at the end of the model with the size equal to the original output size (19) plus the size of the encoded three letter combination array (26 x 26 x 26).

3.5.3 CNN2

In addition to the custom models created for each of the state of the art models above, a self-made CNN model has also been developed by the team, mixing the best practices from known best performing models.

This includes the use of a 3x3 kernel size, popularised by VGGNet, with a stride and padding of 1. This combinations allows for convolution within the image without the loss of output size in the form of width and height. Max pooling has also been used, which is the pooling method used by models such as AlexNet. Lastly, a two layer fully connected neural network is appended at the end which allows for the processing of both the convolutional output as well as the caption encoding, as previously described.

4 Experiment Results

4.1 Module Analysis

In order to determine, the best architecture for this particular problem, the following state of the art as well as custom architectures have been trained and evaluated. The table below demonstrates each model's performance against training and validation datasets. The model with the best mean F1 score will then be examined under Ablation Studies in order to determine components which contribute to its success and other potential improvements.

It was found that AlexNet performed relatively poorly, indicated by the highest training loss lowest F1 score, which was somewhat expected given the other architectures outperformed in the ILSVRC. Upon observation of the outputs, it was seen that the AlexNet models were predicted mostly the same class (1) without any adjustments to weights or loss functions, or the same class of another label (e.g. 3, 5) when weighting methods were applied. AlexNet has a larger filters 11 x 11 relative to the other models, which may cause failure to detect smaller objects in the image, or may have caused general overfitting.
The ResNet models, especially the custom version including captions, performed relatively well which can

be expected due to it’s performance in competitions. This improvement in the ResNet model can be attributed to the extreme depth achieved by the residual mapping and bottleneck layers which can result in greater efficiency.

Models for Evaluation

Sr. No.	Models	Train Loss	Test Loss	F1 Score
1.	CNN2 Kernel: 3 Conv layers: 4 FC layers: 2 Epoch: 3	0.065699	0.002496	0.798142
2.	CNN2 Kernel: 3 Conv layers: 4 FC layers: 3 Epoch: 3	0.070569	0.003528	0.790991
3.	CNN2 Kernel: 3 Conv layers: 5 FC layers: 3 Epoch: 3	0.075042	0.004213	0.783029
4.	AlexNet from Scratch Epoch: 3	0.181809	0.040814	0.428300
4.	AlexNet Pretrained Epoch: 3	0.170711	0.079936	0.463161
4.	Custom Pretrained AlexNet Epoch:			
7.	ResNet Pretrained Epoch: 15	0.033153	0.001527	0.665107
8.	Custom Pretrained ResNet Epoch:15	0.0059	0.00008	0.7826
9.	Inception Pre-trained Epoch:	0.0010	0.00042	0.7916
10.	Custom Pretrained Inception Epoch:	0.0015	0.000011	0.780596

Table 1: Default Parameters for MLP

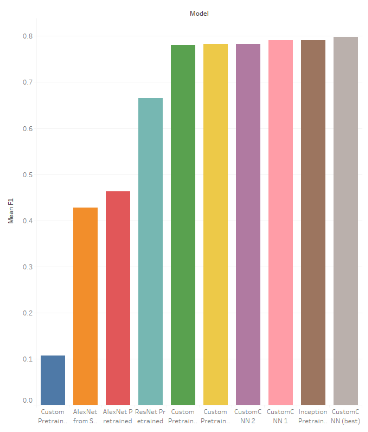


Figure 10: Graphical Analysis for F1 Scores

4.2 Ablation Studies

The table below examines the different impacts in performance, that each component of the best CNN model from the previous section contributes towards. In our study, it was the custom made CNN with 4 convolution layers, each with kernel size of 3, and two fully connected layers that performed the best, and hence will be the subject of this study.

4.2.1 Dropout

The removal of dropout within the fully connected layer was performed in order to see if its regularisation effects does make a difference with model performance. The addition of dropout within neural network models is usually done to prevent overfitting during training by creating the chance of neurons randomly dropping out based on probability.

There is also however, the expectation that having dropout just before the final output layer may negatively impact its ability to make predictions. This is particularly pronounced with the architecture of the CNN2, since it only has two layers, and therefore the presence of dropout between the two may negatively impact it's ability to learn.

As shown within the table, the removal of dropout only marginally reduces the performance of the model. This may reflect that its regularisation is of benefit to the model, however since the difference in performance with and without dropout is quite marginal, there is no evidence that dropout does have a meaningful contribution towards the model.

4.2.2 Captions

The additional data provided by encoding captions into three letter combinations are also tested, with respect to its contribution to the model performance. As previously described within the "Design of Novel Architecture" section, there is expectation that the data provided will be meaningful towards the model as the formation of words present within the caption usually makes a reference to the specified label/s.

The result presented in the table supports this hypothesis, with a 35 percentage point difference between the mean F1 score with captions and without captions. It can therefore be concluded that the use of captions provides meaningful contribution to the model, but also that the methodology in encoding caption data was effective for the model.

4.2.3 Batch Normalisation

Batch normalisation is applied in order to control internal co-variance shift that might occur particularly with deep neural networks. Whilst normalisation is done for the data during pre-processing, without batch normalisation, the data being passed between layers in a neural network may eventually shift beyond the mean that of some of the nodes within the neural network. As a result, this imbalance negatively affects training.

Due to this, the expectation would have been that the inclusion of batch normalisation within the best performing model would have further improved it. However, as per the results table in ablation studies, the introduction of batch normalisation in the convolutional layers have noticeably hampered its performance. It can therefore be concluded, that for this particular problem, batch normalisation does not seem to be of benefit.

Sr. No.	Feature	Train Loss	Validation Loss	Validation Mean F1
1.	Removing dropout within the Fully Connected layer	0.0621	0.0034	0.7914
2.	Removing Captions	0.1604	0.0472	0.4451
3.	Adding Batch Normalisation	0.1226	0.0122	0.5080

Table 2: Performance metrics with each component applied/removed

4.3 Justification on your best model

As described within ablation studies, the best model we conclude for this problem is CNN2, which is the self-made CNN model that we've developed. It seems that the simple model of CNN without batch normalisation, and utilising the learnings of VGG net with a repeated use of kernel size of 3 in convolutions outperforms complicated CNN designs such as Inception and AlexNet.

The reason for such results could have been in the ability to handle of captions data that the self-made CNN2 has, when compared to models ResNet which traditionally does not have a fully connected neural network appended at the end. Or, the specific problem of multi-label classification might require further training or modifications for the state of the art models.

5 Future Scope

Techniques to enhance the visual attention can be implemented to increase the precision of prediction (Lyu et al., 2018). This technique mimics the human way of looking at an image or view, where the first focus is on the whole image and the local specific objects are observed (Lyu et al., 2018).

6 Conclusion

Different model architecture were tried on for the multi-label classification task from which CNN Architecture with three letter Caption Encoding recorded the best performance with average F1 0.798. The Three letter Caption Encoding helped increase the prediction accuracy which concludes the importance of including captions to the overall effectiveness of the model.

APPENDIX

STEPS TO RUN THE CODE SUCCESSFULLY

1. Upload the 500525438_500448953_500393860 - COMP5329 2021S1A2 - Code.ipynb in Googel Colab.
2. Upload the dataset files to a Google Drive Account and paste the link of the 2021s1comp5329assignment2.zip file in the ZIP_PATH field.
3. Run the Libraries and Drive/file/device setup sections and let all files upload completely before running any code cells. There will be a prompt asking for the authentication code which is the access key to the google drive acount.
4. Once all the files are successfully uploaded, run the data read and Miscellaneous Functions sections.
5. The best model for this study CNN2; so scroll down to search the section "Chosen Model Submission" and run the entire section.
6. The final code block under the Chosen Model Submission is the 'Generate submissions export' which creates a file named "submission_file.txt" which contains the final predictions of best model.
7. To run other models architectures, go the specific their section and run the code cells. After executing the code cells, jump to last Section which is 'Generate submissions export' and replace the chosen_model variable according the following table:

Sr. No.	Model in Implementation	Replacement value
1.	Basic CNN	BasicCNN_model
2.	CNN2	CNN2_model
3.	AlexNet from Scratch	ANS_model
4.	Pretrained AlexNet	ANP_model
5.	Custom AlexNet	ANC_model
6.	Pretrained ResNet	RNP_model
7.	Custom ResNet	RNC_model
8.	Pretrained Inception	INCP_model
9.	Custom Inception	INCC_model

Table 3: Model-vise replacement variables for the chosen_model variable in the output$_to_submission()call$

Specifications

The hardware and software specification used to successfully run the code are as follows:

Software
Language: Python3.7 Web-interface: Google Colab Kaggle

Hardware
Windows system: Windows 10
Processor: Intel® Core ™ i7-5500U CPU @2.40GHz 2.40GHz
Installed memory (RAM): 8.00GB
System type: 64-bitOperating System, x64-based processor

References

BCEWithLogisticsLoss. (2021). *Bcewithlogitsloss pytorch 1.8.1 documentation.* Retrieved from `https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html`

Bebnev, V. (2021). *Multi-label image classification with pytorch: Image tagging | learn opencv.* Retrieved from `https://learnopencv.com/multi-label-image-classification-with-pytorch-image-tagging/#:~:text=According\%20to\%20scikit\%2Dlearn\%2C\%20multi,the\%20set\%20of\%20target\%20labels.`

Brownlee, J. (2021). *Convolutional neural network model innovations for image classification.* Retrieved from `https://machinelearningmastery.com/review-of-architectural-innovations-for-convolutional-neural-networks-for-image-classification/`

Canziani, A., Paszke, A., & Culurciello, E. (2017). *An analysis of deep neural network models for practical applications.*

Durand, T., Mehrasa, N., & Mori, G. (2019, June). Learning a deep convnet for multi-label classification with partial labels. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr).*

Lecture 7: Neural netwrok architecture, lecture slides, comp5329: Deep learning, the university of sydney, delivered 26 march 2021. (2021).

Luo, Y., Jiang, M., & Zhao, Q. (2019). Visual attention in multi-label image classification. In *2019 ieee/cvf conference on computer vision and pattern recognition workshops (cvprw)* (p. 820-827). doi: 10.1109/CVPRW.2019.00110

Lyu, F., Hu, F., Sheng, V. S., Wu, Z., Fu, Q., & Fu, B. (2018). Coarse to fine: Multi-label image classification with global/local attention. In *2018 ieee international smart cities conference (isc2)* (p. 1-7). doi: 10.1109/ISC2.2018.8656664

MultilabelSoftMarginLoss. (2021). Retrieved from `https://pytorch.org/docs/stable/generated/torch.nn.MultiLabelSoftMarginLoss.htmlhttps://pytorch.org/docs/stable/generated/torch.nn.MultiLabelSoftMarginLoss.html`

Nelson, D. (2021). *Image classification with transfer learning and pytorch.* Retrieved from `https://stackabuse.com/image-classification-with-transfer-learning-and-pytorch/`

Nigam, V. (2021). *Understanding neural networks. from neuron to rnn, cnn, and deep learning.* Retrieved from `https://medium.com/analytics-vidhya/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90`

scikit learn. (2021). *1.12. multiclass and multioutput algorithms scikit-learn 0.24.2 documentation.* Retrieved from `https://scikit-learn.org/stable/modules/multiclass.html`

Varsheni, S. (2021). *Pytorch transformations | 10 pytorch transformations for data scientists.* Retrieved from `https://www.analyticsvidhya.com/blog/2021/04/10-pytorch-transformations-you-need-to-know/`