COMP5329 Deep Learning Assignment 2

Team members:
- Andrew Zhang (SID:500543568)
- Vincent Yunansan (SID:530454874)

Relevant links:
- Colab notebook (mixed precision) : link
- Result charts (tableau public) : link
- Github repository and readme.md : link

I. Introduction

Background: Deep learning is a collection of general tools which "allows computer models … to learn representations of data with multiple levels of abstractions" (LeCun, Bengio, & Hinton, 2015). Learning is achieved through forward and backward propagations in which internal parameters (e.g. weights and biases) are updated to compute representation of the data as it is propagated through the network, to produce accurate output (e.g. class predictions) at the end. (LeCun, Bengio, & Hinton, 2015). Deep learning techniques "dramatically improved … speech recognition, visual object recognition, object detection, and many other domains" (LeCun, Bengio, & Hinton, 2015).

Study description: This study is part of the COMP5329 Assignment 2 Multi-label Classification Competition 2024 hosted on Kaggle **(Tao, 2024)**. The dataset provided for this study consists of 29,996 and 1,000 images for training and testing. Both training and testing datasets are annotated but only the training dataset is paired with a one or more labels. *Please see Exploratory Data Analysis below for more details.*

Study goal: The aim of this study is to create a robust and fit-for-use deep learning model to solve a multi-label classification problem within the acceptable size (e.g. not larger than 100mb) and training time (e.g. not more than 24 GPU-hours).

Study motivation: Current research in this area favors big and complex architectures which is proficient in the detecting complex features but are mostly overbuilt for our task at hand. As complexity does not determine performance (LeCun, Bengio, & Hinton, 2015), this study explores existing architectures to be adapted for multi-label classification problems and potentially improve upon existing methods.

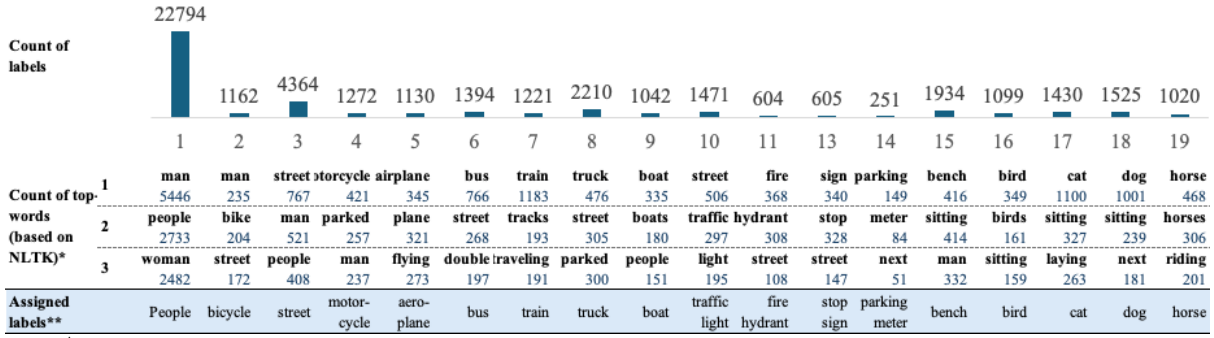
Method overview: Our study is designed to leverage well-proven and established pre-trained model through transfer learning. As explained in chapters below, the use of transfer learning allows us to leverage open-source work, save on computational resources, and minimize uncertainty associated with the trial and error from starting from scratch. In this report, we explore the viability and optimization of multiple pre-trained models and further explore and suggest methods to improve its performance in terms of score and computational efficiency.

Importance of this study: This study will aid in developing deeper understanding of how specific architectures of a model can affect multi-label classification performance in a neural network. Although multi-label classification is becoming more common in the field of deep learning, there is still lack of available methods used in this area (Liu, Wang, Wang, & Tan, 2022). Popular methods such as AlexNet, ResNet, GoogleNet, and VGG were designed for simple image classification tasks (Sharma & Guleria, 2022). Although effective, these methods have not been properly explored for usage in multi-label classification tasks.

Summary of exploratory data analysis: (i) Images have different sizes and label 12 is missing. (ii) We have identified the general meaning of each label by observing common words in the

caption and single labeled images, (iii) There is significant data imbalance (e.g. label 1 appearing the most) and some labels mostly appearing together with other labels (e.g. labels 2, 3, and 8 with high co-occurrence ratios). See **Appendix 1** for sample images and labels and **Figure 1** and **Figure 2** for label statistics.

Figure 1: Count of labels and label name analysis.



Count of top-words (based on NLTK)*	1	man	man	street	motorcycle	airplane	bus	train	truck	boat	street	fire	sign	parking	bench	bird	cat	dog	horse
		5446	235	767	421	345	766	1183	476	335	506	368	340	149	416	349	1100	1001	468
	2	people	bike	man	parked	plane	street	tracks	street	boats	traffic	hydrant	stop	meter	sitting	birds	sitting	sitting	horses
		2733	204	521	257	321	268	193	305	180	297	308	328	84	414	161	327	239	306
	3	woman	street	people	man	flying	double	traveling	parked	people	light	street	street	next	man	sitting	laying	next	riding
		2482	172	408	237	273	197	191	300	151	195	108	147	51	332	159	263	181	201
Assigned labels**		People	bicycle	street	motor-cycle	aero-plane	bus	train	truck	boat	traffic light	fire hydrant	stop sign	parking meter	bench	bird	cat	dog	horse

Note: *Count of top-words accounts for stop-words (e.g. common words not useful for analysis, such as "and", "on", "the") using the Natural Language Tool Kit (NLTK) framework. **assigned labels are based on the team's observation of the NLTK results and on single label examples.

Figure 2: Count of co-occurrence of labels and aggregate statistics.

	People	bicycle	street	motorcycle	aeroplane	bus	train	truck	boat	traffic light	fire hydrant	stop sign	parking meter	bench	bird	cat	dog	horse	co-occurrence total	Total	co-occurrence ratio
People	-	965	3,062	1,003	400	1,062	550	1,454	680	879	245	227	119	1,440	278	255	727	683	14,029	22794	**0.62x**
bicycle	965	-	453	162	7	143	33	160	37	157	41	45	29	118	20	18	60	31	2,479	1162	**2.13x**
street	3,062	453	-	491	136	761	148	1,302	87	910	271	270	165	281	59	57	162	124	8,739	4364	**2.00x**
motorcycle	1,003	162	491	-	2	122	10	242	8	113	15	29	11	40	8	14	49	10	2,329	1272	**1.83x**
aeroplane	400	7	136	2	-	26	-	243	34	14	1	3	-	10	6	2	2	2	888	1130	**0.79x**
bus	1,062	143	761	122	26	-	34	319	20	296	24	45	9	84	16	-	16	31	3,008	1394	**2.16x**
train	550	33	148	10	-	34	-	54	20	147	4	16	1	79	3	1	6	11	1,117	1221	**0.91x**
truck	1,454	160	1,302	242	243	319	54	-	51	378	95	100	41	80	29	6	80	63	4,697	2210	**2.13x**
boat	680	37	87	8	34	20	20	51	-	3	2	3	-	61	77	2	48	17	1,150	1042	**1.10x**
traffic light	879	157	910	113	14	296	147	378	3	-	79	56	32	69	18	1	22	30	3,204	1471	**2.18x**
fire hydrant	245	41	271	15	1	24	4	95	2	79	-	41	11	28	4	3	13	2	879	604	**1.46x**
stop sign	227	45	270	29	3	45	16	100	3	56	41	-	7	11	7	1	3	4	868	605	**1.43x**
parking meter	119	29	165	11	-	9	1	41	-	32	11	7	-	8	3	2	3	1	442	251	**1.76x**
bench	1,440	118	281	40	10	84	79	80	61	69	28	11	8	-	68	52	97	25	2,551	1934	**1.32x**
bird	278	20	59	8	6	16	3	29	77	18	4	7	3	68	-	29	22	14	661	1099	**0.60x**
cat	255	18	57	14	2	-	1	6	2	1	3	1	2	52	29	-	73	3	519	1430	**0.36x**
dog	727	60	162	49	2	16	6	80	48	22	13	3	3	97	22	73	-	35	1,418	1525	**0.93x**
horse	683	31	124	10	2	31	11	63	17	30	2	4	1	25	14	3	35	-	1,086	1020	**1.06x**

Note: co-occurrence ratio signals how often a label is observed with other labels and is calculated as total co-occurrence / total label count in the training dataset.

II. Related works

Classical approach to multi-label image classification: Traditional methods such as decision trees (Tidake & Sane, 2018) and Support Vector Machines (Chen, Bao-Liang, & Kwok, 2006) are usable without significant customization to their architectures but dependent on manual feature selection. This places the users as the bottleneck to performance and scale. Should the user feed the model with suboptimal feature selection, then classification performance would likely suffer.

Use of deep learning in multi-label image classification: Deep learning algorithms, on the other hand, could automatically isolate important features for decision-making (Chauhan & K, 2018). This allows deep learning models to scale independently (i.e. handle much larger multi-label classification problems) as it learns patterns without strict definition provided by the user.

There are limited algorithms specifically developed for multi-label classification: The first deep learning method specifically designed for multi-label classification used a backpropagation Multi-label Learning approach to classify multi-label biological texts (Min-Ling & Zhi-Hua, 2006). The architecture used was a shallow neural network a single hidden layer. The performance of the model stems from its unique error function where it calculated the sum of the exponential of the difference between actual and predicted, instead of a simple sum-of-square error. The architecture yielded the lowest error in evaluation metrics such as hamming loss, one-error, and ranking loss and achieved the highest average precision compared to traditional methods. This method can handle smaller multi-label classification problems but not yet optimized for larger multi-label classification problem such as general images labelling.

A more recent approach for multi-label classification on images utilizes the MT-DNN network (Huang, Wang, Wang, & Tan, 2013). The MT-DNN is deeper than its predecessor (with five hidden layers) and contain multiple binary output nodes for classification. This architecture performed much better compared to traditional methods.

III. Techniques

Principle of methods used in this project are as follows:

- Deep learning: at its most basic form, a deep learning model is a collection of nodes stacked as layers that are connected with each other. The deep learning model propagates information from the input layer all the way to the output layer to create a prediction (also known as forward propagation), calculates how much it misses (with the use of objective or loss function) and update weights and biases at each node according to the gradient of error calculated before (also known as backward propagation). This whole process is repeated until the model has sufficiently learnt the pattern made available and is able to predict patterns based of unseen data (Goodfellow, Bengio, & Courville, 2016).

- Availability of pre-trained models for image classification: pre-trained models trained on single label images are available in open-source machine learning libraries and can be utilized as base model for our multi-label classification exercise. This can be achieved by adjusting the models output to handle multiple labels.

  Figure 3 lists the pre-trained models used for this study. These models are less than 100 megabytes each and were trained on ImageNet, which is a collection of 14.2 million images and 1,000 classes (PyTorch Contributors, 2023)

  Figure 3: pre-trained models used in this exercise

Model	Size (MB)	# params (M)	GFLOPS	ImageNet accuracy score		Docs
				Top-1	Top-5	
GoogLeNet	49.7	6.6	1.50	69.8	89.5	link
ResNet50	97.8	25.6	4.09	76.1	92.9	link
ResNext50_32X4D	95.8	25.0	4.23	77.6	93.7	link
Shufflenetv2_X2_0	28.4	7.4	0.58	76.2	93.0	link
EfficientNet_v2_s	82.7	21.5	8.37	84.2	96.9	link
RegNet_X_3_2_GF	58.8	15.3	3.18	78.4	94.0	link

Note: *GFLOPS: billion floating point operations per second, is a measure of model's calculation complexity. Higher GFLOPS generally means higher computational loads.

- GoogLeNet (2014): developed by Google, this architecture is able to achieve significantly more depth (22 layers) compared to its predecessors at manageable computational costs by introducing (i) 'inception modules' which consists of 1x1, 3x3, and 5x5 convolutional filters and 3x3 pooling operations. Inception modules allow the network to 'learn and choose' from different sized filter in each layer, (ii) 'auxiliary classifiers' which are attached to intermediate layers. Auxiliary classifiers contribute weighted loss values from intermediate layers to the final loss calculation to address the vanishing gradient problem (Szegedy, et al., 2014).

Figure 4: Naïve Inception Module (Szegedy et al., 2014)

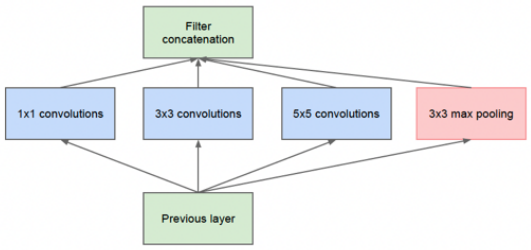
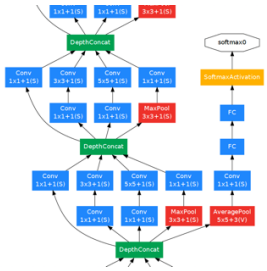


Figure 5: auxiliary classifiers in yellow (Szegedy et al., 2014)



- ResNet (2015): introduced the use of 'residual blocks' and 'shortcut connections'. Shortcut connections address the vanishing gradient problem by allowing input of an earlier layer to 'skip' one or more layers to become input of a deeper layer. This technique allows for much deeper networks, reaching 152 layers for ImageNet (He, Zhang, Ren, & Sun, 2015).

Figure 6: Residual blocks with shortcut connections (He et al., 2016)

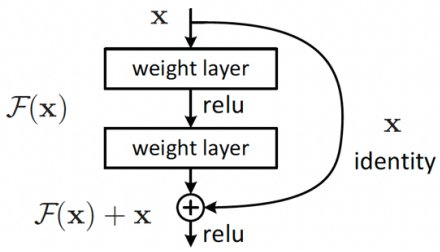
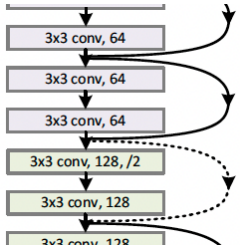


Figure 7: Skip connections propagate input to next layer (He et al., 2016)



- Resnext50 (2016): is an extension to ResNet which introduces additional branches called 'cardinality'. Cardinality splits layers into multiple pathways, allowing models to scale up in a new dimension outside of depth and width, and aggregates the output at the end of the block. This technique is useful in expanding a model while maintaining complexity as model growth is 'spread out' to other dimensions (Xie, Girshick, Dollar, Tu, & He, 2016).

Figure 8: cardinality introduces parallel pathways (Xie et al., 2016)

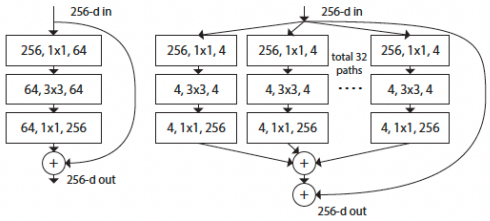
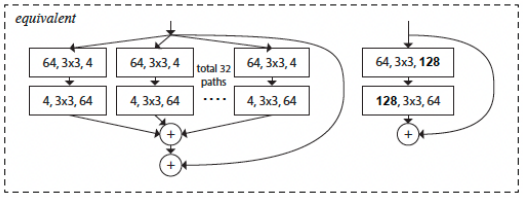
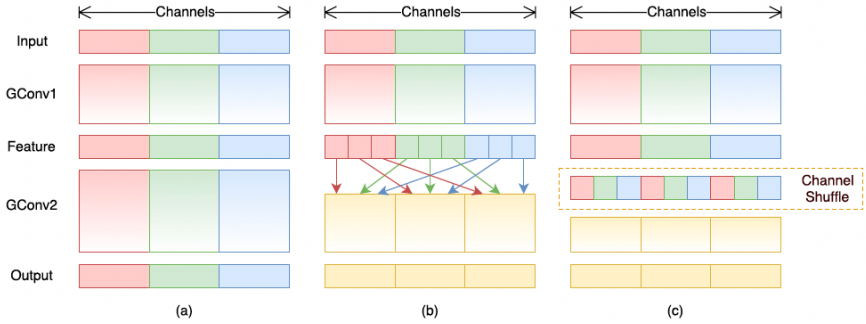


Figure 9: cardinality allows for narrower network in a single branch (Xie et al., 2016)



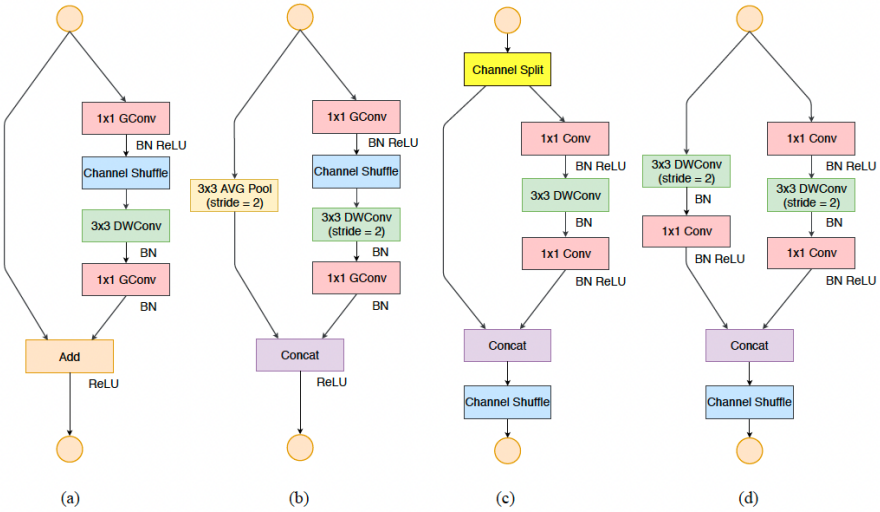
- ShuffleNetV2 (2018): first developed in 2017 for "mobile devices with very limited computing power", ShuffleNet utilizes (i) 'group convolutions' which convolves the input to reduce the number of parameters and (ii) 'channel shuffle' which allow group convolutions in the next layer to receive input from all preceding groups (Zhang, Zhou, Lin, & Sun, 2017). ShuffleNetV2 is an iteration of ShuffleNet which introduces channel splitting and simplified design (Ma, Zhang, Zheng, & Sun, 2018).

Figure 10: Channel shuffle with two stacked group convolutions (Zhang et al., 2017)



Note: a) no channel shuffle; b) group convolutions take data from different groups after preceding group c) implementation of channel shuffle which achieves the same outcome as b).

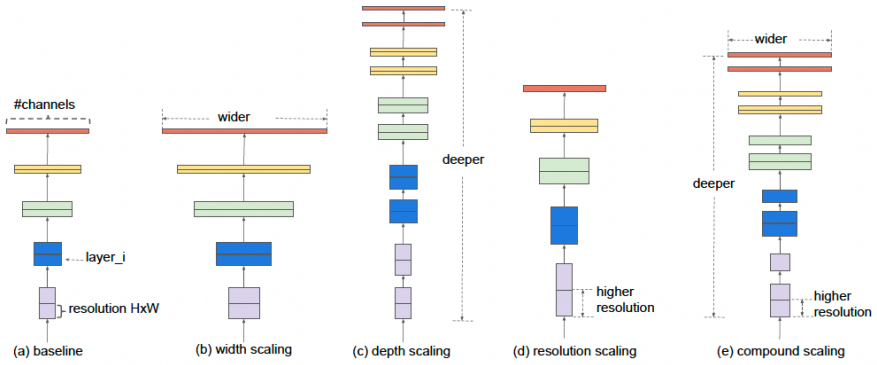
Figure 11: Architecture of ShuffleNet and ShuffleNetV2 (Ma et al., 2018)



Note: a) basic ShuffleNet; b) ShuffleNet unit for spatial down-sampling c) ShuffleNetV2 d) ShuffleNetV2 for spatial down-sampling.

- EfficientNetV2(2021): developed by Google Research, EfficientNet is a family of models developed under the idea of a 'compound coefficient' which systemically scales depth/width/resolution (Tan & Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 2020). The main building block of EfficientNet is the 'mobile inverted bottleneck' where (i) input is expanded, (ii) passed through depth-wise convolution to create non-linearity, and (iii) projected back to a lower dimension (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018). EfficientNetV2 is an update to the EfficientNet family of models with "faster training times and better parameter efficiency (Tan & Le, EfficientNetV2: Smaller Models and Faster Training, 2021)

Figure 12: Compound scaling under EfficientNet (Tan & Le, 2019)



- RegNet (2020): developed by Facebook AI Research (FAIR), RegNet is based on the idea that model design parameters should be "simple, work well, and generalize across settings". Experiments run by FAIR displayed clear quantized linear relationship between network parameters, such as depth and width. The authors of RegNet claims that "Regnet is able to achieve good results with simple models" and RegNet is able to "outperform EfficientNet by 5x on GPUs" (Radosavovic, Kosaraju, Girshick, He, & Dollar, 2020). *Regnet to be discussed more in Chapter IV. Experiments and Results.*

- Transfer learning and adjustments applied to the pre-trained model: We applied minimal changes to the pre-trained models to leverage their trained weights and biases. The team believes layers within the models function in tandem and any significant changes to the models' architecture may result significant re-training requirements.

  To adjust the pre-trained models to handle multi-label classification problems, the output layer had to be adjusted to produce multi-label prediction of 18 labels, instead of a single prediction. Binary Cross Entropy with Logits (BCE with logits) was used as criterion in this multi-label classification problem. BCE with logits is two-step calculation where probability (between 0 and 1) of a label is calculated and binarized through a sigmoid hurdle (e.g. output is 1 if probability is higher or equal to hurdle and 0 otherwise) (PyTorch Contributors, 2023). To rephrase, the final layer produces an output matrix which has width equivalent to the number of classes (e.g. 18 classes for our project) and each cell in the output matrix yields 1 (True) for any classes that are predicted to be in the image and 0 (False) otherwise.

  Justification and advantages of transfer learning: the use of pre-trained model removes the uncertainty from trial and error associated with building a model from scratch. The models chosen are of known track record against a large image dataset and known size. The use of

pre-trained model also saves on computational resources as internal parameters already learnt the features of objects in an image.

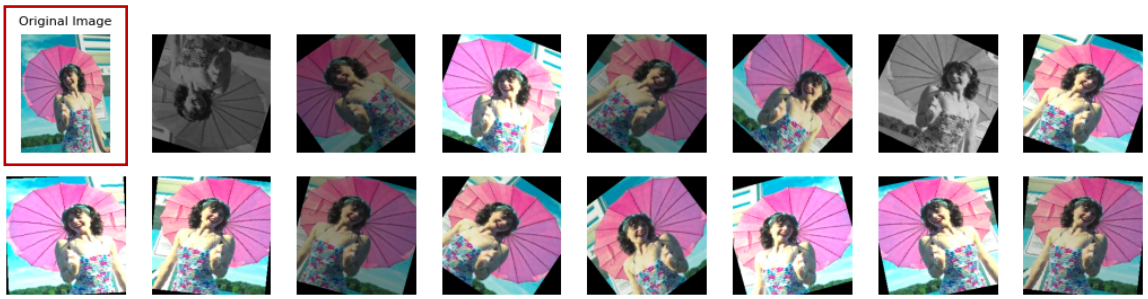
- Mini-batch allows users to train and update the model's parameters with more than one sample at a time for higher training speeds and smoother parameter updates. The implementation of mini batch can be adjusted to match expected granularity of parameter updates and/or device limitations (e.g. a small laptop with no GPU can handle much smaller batches compared to a cloud super-computer) (Brownlee, 2019).

- Data augmentation introduces noise to the dataset to train models with better generalizability. Data augmentation is especially important for image / object recognition as images have a high range of variation in real life, thus an image recognition algorithm has to be able to learn noisy patterns (Goodfellow, Bengio, & Courville, 2016).

  The following transformation was applied to the training dataset:
  - Images were resized to (256x256) to create uniform input shape for the model.
  - Images were then varied through random cropping (224x224), colour jitter (e.g. random changes in brightness), rotation, horizontal & vertical flips, and random grey-scaling.
  - Images were then transformed into tensors and normalized.

  The validation and testing datasets were resized to (256x256), centre-cropped to (224x224), transformed into tensors and normalized.

  Figure 13: Transformations applied to image 0.jpg in the training dataset.



- Optimizers: several optimization methods are used to minimize the loss function efficiently. The optimizer vary in their 'strengths' to create variety in model aggressiveness. The optimizers used in this exercise are as follows:

  - Stochastic Gradient Descent (SGD): defines the gradient as an expectation and follows the estimated gradient 'downhill'. SGD allows updates of parameters with small batches of the dataset instead of using the entire dataset in a single epoch which may be prohibitive for large datasets (Goodfellow, Bengio, & Courville, 2016).

  - AdaDelta: is an optimizer with adaptive learning rate which adjusts learning rates of parameters during the training process. AdaDelta was developed to address hyper-aggressive learning rates in AdaGrad as it "accumulates squared gradients from the beginning of training" which may impact the model's ability to converge (Goodfellow, Bengio, & Courville, 2016).

- Adam: is another optimizer with adaptive learning rate which combines the capabilities of RMSProp and AdaGrad by implementing the first and second moments to adapt the learning rate (Goodfellow, Bengio, & Courville, 2016). Adam could converge quicker than other optimizers and is resistant to noise in dataset by implementing a correction factor in calculating the updates to the first and second moments (Wei, 2024).

- Train and validation split, dynamic learning rate, and early stopping were implemented to ensure models did not overfit and save on compute resources.

- Mixed precision: is an implementation technique available for NVIDIA GPU users in the PyTorch library which allows the usage of lower precision floating point in calculations where appropriate. This results in faster compute time at the cost of performance (PyTorch Contributors, 2023).

IV. Experiments and results

Due to the team's hardware limitation, Experiment 1 (Transfer Learning) is done on MPS (Apple M2 Pro chip) and Experiment 2 (Mixed Precision on best model) is run on a shared CUDA (NVIDIA GeForce RTX 3090) environment.

Relevant links:
- Colab notebook (mixed precision) : link
- Result charts (tableau public) : link
- Github repository and readme.md : link

A. PROJECT STRATEGY AND DESIGN

This project utilizes transfer learning whereby models trained on ImageNet are used as the starting point for a model to solve our task. This technique allows for certainty of usability (e.g. we have proof from the start that the pre-trained models performs well on image classification problems) and significant savings in training resources (e.g. we do not need to train the model from scratch).

All pre-trained models used are less than 100 megabytes to adhere with the project specification and are part of an open-source library (PyTorch Contributors, 2023). To further accelerate training time, the team also implemented dynamic learning rate, early stopping, mini-batch training, and a range of criterions to fine the best mix which yields the best outcome.

The wide range of models and optimizers are intended to provide a variety of model/criterion complexity/sophistications and identify the 'sweet spot' of strong performance (e.g. F1-score beyond the hurdle and minimize overfitting) and acceptable training costs (e.g. under 24 GPU-hours).

The team then suggests one best model and mixed precision training to analyse the trade-off between saved computational cost and lost performance (PyTorch Contributors, 2023). Other improvement ideas that are not implemented in this project are discussed in Chapter V: Conclusion and Discussion.

EXPERIMENT SET UP:

- Training and testing dataset are loaded onto DataLoaders with the appropriate transformations as explained above. The training set is further split into training and validation sets to monitor for overfitting.

- Experiment 1 (Transfer Learning): Hyperparameter testing is done on the 6 pre-trained models and 3 optimizers described above. The maximum number of epochs is set at 60 epochs to identify which models can converge quicker and provide better results within acceptable computation times. Logs, prediction output, and saved model are automatically generated and saved. Logs include measures such as epoch count, training time of the epoch, training and validation loss scores, and F1-score at validation.

  Note on F1-scores: The team measured macro-average F1-score at validation as it highlights performance of under-represented classes in this dataset. In other words, the validation F1 score takes an arithmetic average of F1 score of each class, regardless of their saliency in the dataset, creating a fair representation of precision and recall of an imbalanced dataset (Leung, 2022). Per our observation, (validation) macro-average F1 score of 70.0%+ generally translates to (test) micro-average F1 score of 88.5%+.

  As explained in the previous chapter, minimal change was applied to the models' architecture. The team adjusted the fully connected layer of each model to accommodate multi-label classification of 18 classes (as opposed to single-classification of 1000 classes of the original ImageNet pre-trained models) but did no other adjustments to the model architecture. The team believes that deviating too much from the original architecture (e.g. by trimming down the layers to create a smaller model) would alter the model too much and would require significant training times to re-adjust the weights, defeating the whole purpose of transfer learning on models already smaller than 100 megabytes.

- Experiment 2 (Mixed Precision on Best Model): the best model identified from the first experiment and is re-trained with mixed precision for 80 epochs to analyse if (i) the time saved with lower precision results in significantly worse performance and if (ii) the lost performance can be recouped with additional training.

B. DEFINITION OF BEST MODEL

We define our best models with (test) micro-average F1 scores equal or higher than 88.5% within acceptable size and training time parameters. The F1 score calculation is specified in the project documentation as:

$$F1 = 2\frac{p * r}{p + r} \quad \text{where} \quad p = \frac{true\ positive}{true\ positive + false\ positive} \quad r = \frac{true\ positive}{true\ positive + false\ negative}$$

Source: *Multi-label Classification Competition 2024* (2024)

C. PROJECT RESULTS AND DISCUSSIONS

o Results and Observations from Experiment 1 (Transfer Learning):

- Model performance:

  - GoogleNet and ShuffleNet (6.6 and 7.4 million parameters each) underperform ResNet, ResNext, and RegNet (25.6, 25.0, 15.3 million parameters each). We hypothesize that GoogleNet and ShuffleNet are not converging fast enough due to

their more simple architectures and will require higher epoch counts to get to equitable results.

- EfficientNet (82.7 million) failed to converge under multiple training attempts and was abandoned. We observed significant instability where a single epoch may run between 8 minutes to 500 minutes per epoch or not converging at all. Although EfficientNet is not the largest model by number of parameters, it is much heavier in terms of computational load at 8.37 GFLOPS, almost twice as heavy as the next model in our collection.

- Optimizer performance: Adam is better than AdaDelta and SGD in under-performing models (e.g. GoogleNet and ShuffleNet) and overfits in stronger models (e.g. ResNet, ResNext, and RegNet). This suggests a 'sweet spot' between model vs. optimizer strengths.

- Validation and Test F1 Scores: ResNet, RegNet, and ResNext achieved close to or above 0.700 of (validation) macro-f1 score and were submitted to the Kaggle platform. (Test) micro-F1 score were similar at around 0.900.

- Best performers: models produced by ResNet, ResNext, and RegNet passed our performance hurdle and achieved similar (test) micro-average F1 scores with the following highlights: (i) ResNet provided the best 'bang for buck' in terms of performance for training time, (ii) RegNet was best in terms of performance for size.

  Both ResNet and RegNet models were trained well within the 24 GPU-hour limit. As such, we defined RegNet trained with AdaDelta as our best model. See results below:

Figure 14: validation and test F1 scores.

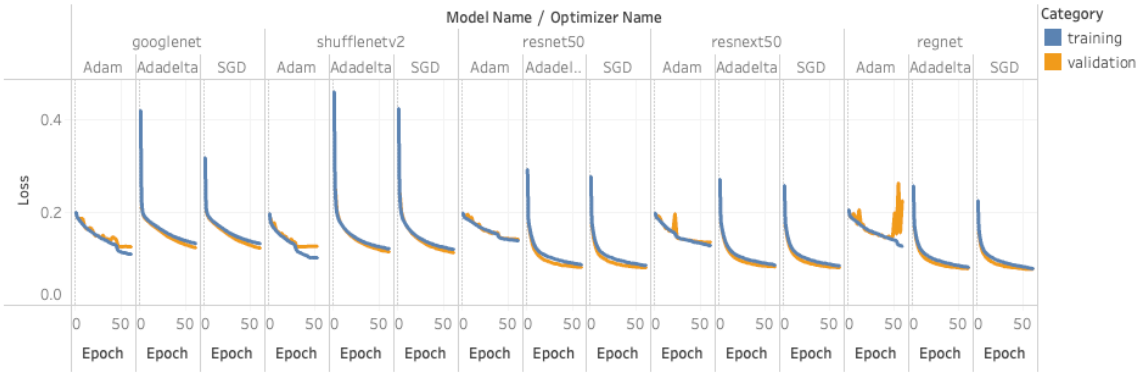
	val macro - F1			test micro-F1		
	Adam	AdaDelta	SGD	Adam	AdaDelta	SGD
EfficientNet	failed to converge			Excluded from test submission due to inferior val score		
GoogleNet	0.466	0.353	0.352			
ShuffleNet	0.508	0.458	0.468			
ResNet	0.362	0.708	0.720		0.895	0.895
RegNet	0.415	0.727	0.733		0.900	0.898
ResNext	0.399	0.697	0.707		0.897	0.897

	training time (hours)			val macro F1 / training time			test micro F1 / training time		
	Adam	AdaDelta	SGD	Adam	AdaDelta	SGD	Adam	AdaDelta	SGD
EfficientNet							Best performance for training time		
GoogleNet	3.89	4.22	3.58	0.12	0.08	0.10			
ShuffleNet	2.74	4.83	3.16	0.19	0.09	0.15			
ResNet	6.42	6.39	6.27	0.06	0.11	0.11		0.14	0.14
RegNet	8.94	9.02	6.80	0.05	0.08	0.11		0.10	0.13
ResNext	8.11	8.52	7.77	0.05	0.08	0.09		0.11	0.12

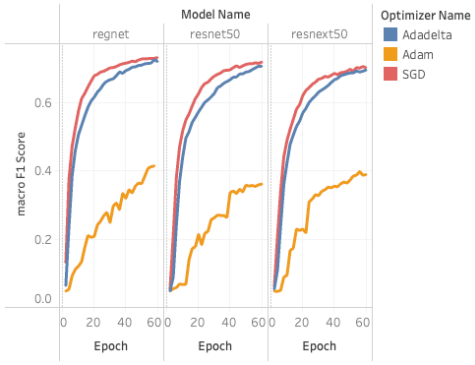
	Size (MB)	val macro F1 / size * 100			test micro F1 / size * 100		
		Adam	AdaDelta	SGD	Adam	AdaDelta	SGD
EfficientNet	82.70				Best performance for size		
GoogleNet	21.60	2.16	1.63	1.63			
ShuffleNet	20.76	2.45	2.21	2.25			
ResNet	90.12	0.40	0.79	0.80		0.99	0.99
RegNet	54.98	0.75	1.32	1.33		1.64	1.63
ResNext	88.16	0.45	0.79	0.80		1.02	1.02

Figure 15: Training and validation loss values by epoch

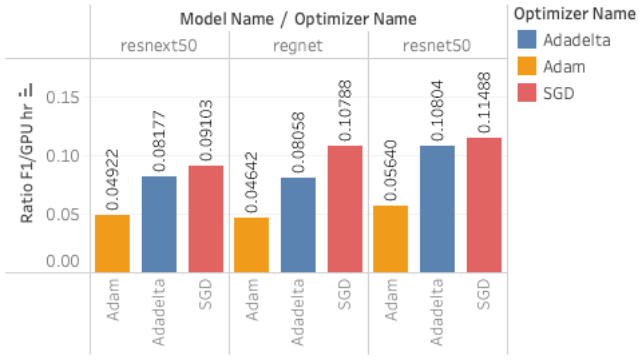
Training and Validation Loss (sorted by training loss)



Macro F1 score on validation data



Ratio of Macro F1 score / Training time (GPU-hours)



Further discussion on RegNet: RegNet is a deep learning network that is built upon ResNet (Xu, et al., 2022). The architecture of the RegNet includes skip connections and regulatory modules using Convolutional Recurrent Neural Networks ('Conv RNNs'). Just like the typical RNNs in deep learning, Conv RNNs utilize a hidden state to serve as a checkpoint to save spatiotemporal information of the image (Xu, et al., 2022). This hidden state is constantly updated and used in the subsequent convolutions, which retains information in deeper convolutional layers.

This theory was reflected in a comparison in the feature maps between RegNet and ResNet, where it was found that RegNet preserved more subtle features learned in the early layers due to its hidden state and performed better than ResNet in classification.

To showcase the inner functions of our RegNet model (and other deep learning models in general), we fed forward a single example through our RegNet model and visualized feature maps at different layers of the model (see **Figure 16**) and visualized a heatmap with grad-cam to highlight which feature areas are processed by the model in coming up with a classification prediction (see **Figure 17**).

By observing the feature map in **Figure 16**, we can see low-level features extracted in earlier layers becoming more complex/abstract as the layers went deeper. The abstraction in deeper

layers is due to sequential activations between layers, down-sampling (which lowered spatial resolution), and skip connections between alternate layers (which passes complexity to deeper layers to solve the vanishing gradient problem). This abstraction represents learning in a deep learning model.

The heatmap in **Figure 17** shows the model identifying regions of the image that carries relevant information for class identification. Specific to the cat image used for this analysis, we observe how the model is able to differentiate the cat's facial feature (high heatmap value, in red) against its body (neutral heatmap value, in green) and backdrop (lower heatmap value, in teal).

Figure 16: Representation of feature map in different layers, notice the abstraction in deeper layers

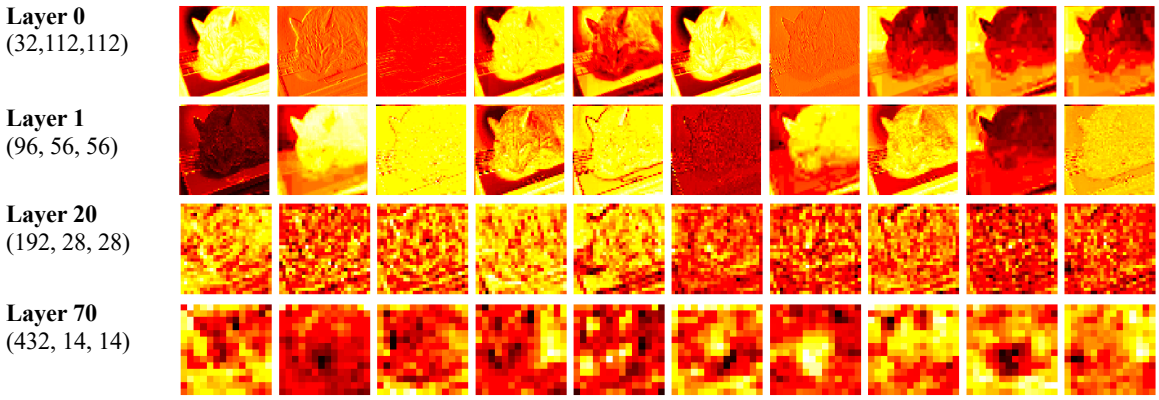
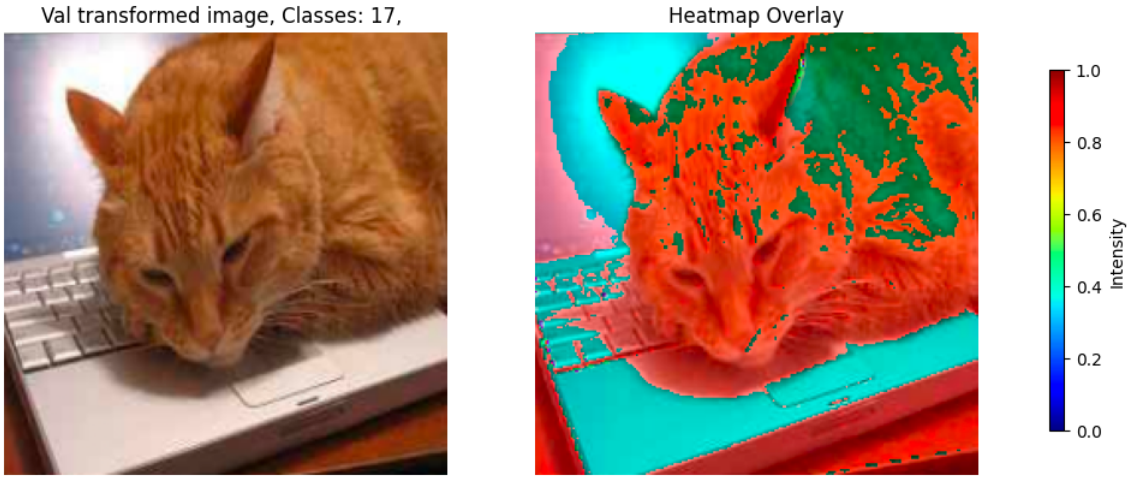


Figure 17: input image and feature heatmap, notice features extracted by the model in red



- Results from Experiment 2: Trade-off between Precision and Speed

Due to limited access to a (shared) cuda GPU, the team was only able to run a single model to test for the impact of mixed precision. The team opted to use the best model on for this experiment with results presented below.

- Base model: the best model from Experiment 1 (RegNet with AdaDelta) was used as the base model for Experiment 2 (Mixed Precision Training).

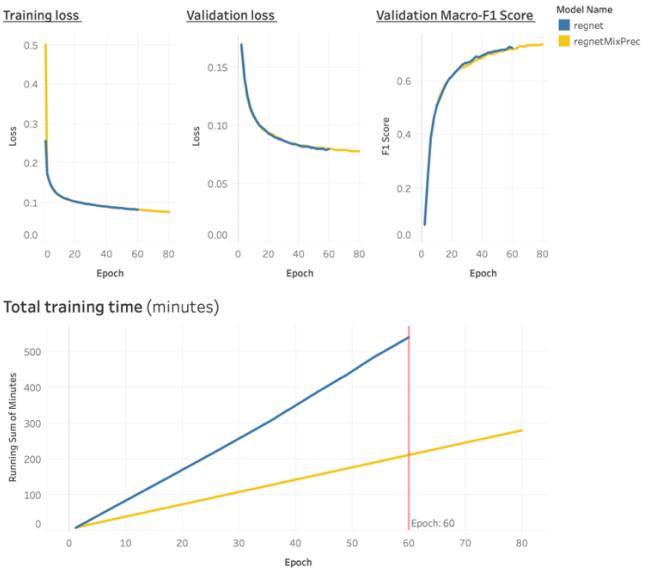
- Experiment outcome: mixed precision resulted in c.60% training time savings (at 60 epochs) with only 3 basis points of deterioration to the (validation) macro-average F1 score. Further training increased the (validation) macro-average F1 well beyond the previous best and (test) micro-average F1 scores are comparable (0.895 vs. 0.900).

- Best model: We define RegNet with AdaDelta and Mixed Precision as our best model as it satisfies our technical criteria in a significantly shorter amount of time.

Figure 18: validation and test F1 scores.

		Val Macro F1	Test Micro F1
		AdaDelta	Adam
RegNet (60 epochs)		0.722	0.900
RegNet w/ Mixed Precision (60 epochs)		0.719	
RegNet w/ Mixed Precision (80 epochs)		0.736	0.895
	Training time	**Val Macro F1 / Training time**	**test micro F1 / training time**
RegNet (60 epochs)	9.02	0.08	0.100
RegNet w/ Mixed Precision (60 epochs)	3.54	0.20	
RegNet w/ Mixed Precision (80 epochs)	4.69	0.16	0.191
	Size (MB)	**Val Macro F1 / Size * 100**	**test micro F1 / size * 100**
RegNet (60 epochs)	54.98	1.31	1.64
RegNet w/ Mixed Precision (60 epochs)	54.99	1.31	
RegNet w/ Mixed Precision (80 epochs)	54.99	1.34	1.63

Figure 19: Mixed Precision vs. Standard RegNet Scores and Training Time



V. Conclusion and Discussion

Our study highlighted the potency of transfer learning and its application in multi-label classification. The usage of pre-trained models from the ImageNet database allows weights to be saved and loaded into our model implementation (Marques, Gois, Madeiro, Li, & Fong, 2022). This meant that the weights contained information on the 14 million images it was initially trained under, providing sufficient information on pixel intensities, textures, and patterns. The preloaded information subsequently provides an advantage over creating a model from scratch as it had initial knowledge to start off with. Additionally, our dataset contains missing and unbalanced labels, and transfer learning with a model trained under the ImageNet database likely contains these images that could circumvent this problem (Marques, Gois, Madeiro, Li, & Fong, 2022).

Furthermore, transfer learning allows for a higher learning rate. As the model was likely already trained on similar tasks, the learning rate in fine-tuning the model could be adjusted higher, which increases the computational efficiency of training.

However, a potential limitation in transfer learning is negative transfer learning where using a pre-trained model could result in a worse performance instead. This phenomenon was overcome using data augmentation and hyperparameter optimizations.

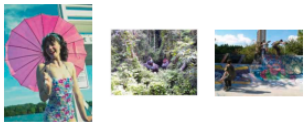
Ultimately, transfer learning is an excellent framework to create a robust model but not without its limitations. Careful implementation of data preprocessing and evaluation techniques is required to ensure the development of a generalizable model for multi-label classifications.

Next development ideas: Future research to improve our framework includes creating a more robust evaluation implementation, such as cross-validation (to ensure convergence and generalizability), parallel ensemble learning (e.g. implementing a LSTM model to process image captions in parallel to the image classification model and combining their output), and/or sequential ensemble learning (e.g. combining a bigger pre-trained base model with a smaller boosting model in a sequential ensemble structure to correct errors) (Mohammed & Kora, 2023).

VI. Appendix

Appendix 1: Single label sample images

1 (people)



7 (train)

14 (parking meter)

2 (bicycle)

8 (truck)

15 (bench)

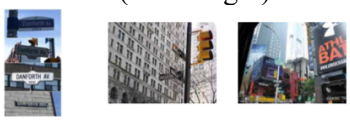
3 (street)

9 (boat)

16 (bird)

4 (motor-cycle)

10 (traffic light)



17 (cat)

5 (aero-plane)

11 (fire hydrant)

18 (dog)

6 (bus)

13 (stop sign)

19 (horse)

VII. References

Brownlee, J. (2019, August 19). *Machine Learning Mastery*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

Chauhan, N. K., & K, S. (2018). A Review on Conventional Machine Learning vs Deep Learning. *2018 International Conference on Computing, Power and COmmunication Technologies (GUCON).*

Chen, K., Bao-Liang, L., & Kwok, J. T. (2006). Efficient Classification of Multi-Label and Imbalanced Data using Min-Max Modular Classifiers. *The 2006 IEEE International Joint Conference on Neural Network Proceedings.*

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv*, 1512.03385.

Huang, Y., Wang, W., Wang, L., & Tan, T. (2013). Multi-task deep neural network for multi-label learning. *2013 IEEE International Conference on Image Processing.*

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, 436-444.

Leung, K. (2022, January 5). *Towards Data Science*. Retrieved from Towards Data Science: https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Liu, W., Wang, W., Wang, L., & Tan, T. (2022). The Emerging Trends of Multi-Label Learning. *IEEE Transactions oon Pattern Analysis and Machine Intelligence*.

Ma, N., Zhang, X., Zheng, H.-T., & Sun, J. (2018). ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. *arXiv*, 1807.11164v1.

Marques, J. A., Gois, F. N., Madeiro, J. P., Li, T., & Fong, S. J. (2022). Chapter 4 - Artificial neural network-based approaches for computer-aided disease diagnosis and treatment. *Cognitive and Soft Computing Techniques for the Analysis of Healthcare Data*, 79-99.

Min-Ling, Z., & Zhi-Hua, Z. (2006). Multilabel Neural Networks with Applications to functional Genomics and Text Categorization. *IEEE Transactions on Knowledge and Data Engineering, 18(10)*, 1338-1351.

Mohammed, A., & Kora, R. (2023). A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University - COmputer and Information Sciences. Volume 35, Issue 2*, 757-774.

Multi-label Classification Competition 2024. (2024). Retrieved from Kaggle: https://kaggle.com/competitions/multi-label-classification-competition-2024

PyTorch Contributors. (2023). *Automatic Mixed Precision package - torch.amp* . Retrieved from Pytorch: https://pytorch.org/docs/stable/amp.html

PyTorch Contributors. (2023). *Pytorch.org*. Retrieved from Pytorch.org: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

Radosavovic, I., Kosaraju, R., Girshick, R., He, K., & Dollar, P. (2020). Designing Network Design Spaces. *arXiv*, 2003.13678v1.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Sharma, S., & Guleria, K. (2022). Deep learning Models for Image Classification: Comparison and Applications. *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering(ICACITE)*.

Szegedy, C., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., . . . Rabinovich, A. (2014). Going Deeper with Convolutions. *arXiv*, 1409.4842.

Tan, M., & Le, Q. V. (2020). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv*, 1905.1194v5.

Tan, M., & Le, Q. V. (2021). EfficientNetV2: Smaller Models and Faster Training. *arXiv*, 2104.00298v3.

Tao, L. (2024). *Kaggle*. Retrieved from Kaggle: https://www.kaggle.com/competitions/multi-label-classification-competition-2024

Tidake, V. S., & Sane, S. S. (2018). Evaluation of Multi-label Classifiers in Various Domains Using Decision Tree. *Intelligent Computing and Information and Communication*.

Wei, D. (2024, April 5). *Medium*. Retrieved from Medium: https://medium.com/@weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e#:~:text=Here's%20why%20Adam%20has%20become,Stochastic%20Gradient%20Descent%20(SGD).

Xie, S., Girshick, R., Dollar, P., Tu, Z., & He, K. (2016). Aggregated Residual Transformations for Deep Neural Networks. *arXiv*, 1611.05431.

Xu, J., YuPan, Pan, X., Hoi, S., Yi, Z., & Xu, Z. (2022). RegNet: Self-Regulated Network for Image Classification. *IEEE Transactions on Neural Networks and Learning Systems.*

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2017). ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *arXiv*, 1707.01083v2.