https://github.com/sazzouz/AMLS_20-21_SN15043735

APPLIED MACHINE LEARNING SYSTEM ELEC0132 20/21 REPORT

SN: 15043735

ABSTRACT

This report explores analysis of deep learning techniques for a well-known computer vision challenge known as image classification. The classification tasks are split into binary and multi-class task, within which two sub-tasks involve a different region of interest (ROI) that is expected to be classified. Facial-feature ROI techniques such as facial landmark extraction and Haar cascade classifiers have been utilized to capture the most important details in the data. The models evaluated in each task build upon the convolutional neural network (CNN) architecture. In the interest of the proliferation of connected systems will have on society, mobile-oriented pre-trained networks are utilized to illustrate their performance boosting capabilities whilst being optimized to be deployed on low-power embedded systems where resources efficiency and latency are the utmost priorities in most applications. The same models are used in each task, where these models are tuned separately for each challenge and dataset, thus providing a more comprehensive analysis as well as a consistent investigatory thread throughout the report

1. INTRODUCTION

The evolution of computer vision in recent times has enabled the development of a plethora of highly impactful and disruptive technologies in a broad range of industries. Simultaneously, the increased accessibility in the development of internet-connect embedded systems, commonly referred to as the 'Internet-of-Things', has enabled computer vision to find applications in a diverse range of settings, such as agriculture, industrial processes, and UAV drones. Traditionally, the use of deep learning in embedded systems was largely prohibitive due to the hardware demands. However, with continual advancements being made in the processing power of small-scale devices, deep learning networks can now be deployed to run more optimally on mobile devices. Such networks are often referred to as ‘MobileNets’ [1], where these models have been developed to performantly handle the inevitable tradeoff between network complexity and resource requirements during training and inference processes. As IoT and embedded systems are expected to be deployed at the edge, often latency is prioritized over peak accuracy, as such there ought to be a optimal middle ground which is capable of being deployed on such devices whilst honoring the expectations set in terms of latency and convergence time.

Inspired by the potential impact of embedded computer vision applications on society, this report includes a step-by-step discussion in tackling each of the four tasks outlined and the methodologies used to converge to best-performing solutions following experimental analysis. A consistent pipeline framework was utilized for each task which included the following activities: (1) Data Preprocessing, (2) Image Augmentation, (3) Tuning, (4) Training and (5) Evaluation. As this report serves as an ablation study, multiple models, each of varying complexity, are tuned and trained to draw conclusions about the best performing and most suitable mobile-oriented deep neural network for each of the image classification tasks, whereby interesting insights can be drawn about the tradeoff that adding complexity may inevitably bring, and also in keeping with not ‘No Free Lunch’ theorem whereby not one model will fit all scenarios, therefore we must explore alternative solutions.

This report is organized as follows: In the next section, relevant theoretical concepts drawn from literature are discussed; Section 3 outlines the model topologies used throughout the report; Section 4 covers the implementation pipelines used in solving each task; Section 5 discusses specialized implementation details for each task and finally Section 6 provides analysis of the experimental results.

2. LITERATURE SURVEY

2.1 – Haars Cascade Classifiers

Haar Cascade Classifiers [3] implement an object detection method proposed by Paul Viola and Michael Jones, often referred to as the Viola-Jones method, in their paper “Rapid Object Detection using a Boosted Cascade of Simple Features” at the turn of the century. This machine learning algorithm is trained on a series of 'positive' and 'negative' images, that is images that contain and do not contain the desired feature, respectively. To extract ROIs from the images provided, the algorithm primarily utilizes unitary features such as edge, line, and four-rectangle features to progressively calculate the sum of pixels under white and black rectangles from each part of the image. These quantities are then fed into the Adaboost algorithm for classification to occur This technique has been shown to be effective at training models to detect and classify facial features in humans.

2.2 – Histogram of Oriented Gradients (HOG)

This report utilizes an implementation of the HOG [2] technique provided by the dlib library [12] accompanied by a 68 point facial point annotation predictor [8], providing essential accessories to the HOG algorithm for use in facial feature applications such as a linear classifier, image pyramid and sliding window detection scheme. Although capable of performing as a general-purpose object detector, HOG has been shown to be highly effective at detecting human faces. This algorithm is robust for object detection applications because the shapes of objects within images is depicted using the localized edge direction and intensity of gradient distributions.

2.3 – Convolutional Neural Networks

The Convolution Neural Network (CNN) [4] has been routinely shown to be a highly effective deep learning architecture for using in image classification problem. Inspired by the neuron patterns found in the visual cortex of the human brain, CNNs includes layer connectivity that responds to stimuli induced by the presence by features learned from input images. An important feature that has proven CNNs to be superior architectures for image classification problems in comparison to traditional feed forward networks such as multi-layer perceptron's is the ability to capture both spatial and temporal dependencies found within images through the successive application of appropriate filters, enabling CNNs to build a more sophisticated understanding of the target images throughout the training process. Central to the architecture of CNNs is the convolutional kernel layer which enables this type of neural network to reduce input images into a form that is easier to process without sacrificing the features that are most crucial in providing accurate predictions. Convolutional filters progressively move throughout a given image at a rate managed by the stride length attribute until the whole image has been traverse. As a result, low-level features such as edges are extracted from the input image and may be fed to subsequent convolutional layers which are then capable of contributed a progressively high-level representation of the original input. The additional complexity introduced by additional convolutional layers results in a higher-detail depiction of the original input.

To perform as an image classifier, the CNN's are often terminated with a flattening process so that the latent image data can be ingested into a conventional into feed forward neural networks for classification, often implemented with a high-neuron fully connected layer shortly followed by an output layer with several dense neuron units equivalent to the number of desired classes.

.

2.4 – Transfer Learning

A method known as transfer learning [5] is known to boost the performance of a simple convolutional neural network on small datasets is to harness feature-extracting capability of larger networks that have been pretrained on a broader and more diversified dataset. This concept is often implemented by 'freezing' the base convolutional blocks within a pretrained network except for the final convolutional block, terminated with a fully connected classifier network. This is made possible through small incremental weight updates to the last step in the convolutional block that enables classification to be performed on new datasets. Only fine-tuning the final convolutional block helps to prevent overfitting on the new dataset which is made more likely due to the greater entropic capacity of the pretrained base network. This is in part due to the nature of convolutional networks in learning progressively higher-level features of input images, whereby the lower-level features extracted are likely to be reasonably generic and could be applied to new datasets. Then, the fine-tuned final convolutional block is used to specialize the classification capabilities of the overall model. This report explores the performance of 'MobileNets', pretrained deep neural networks capable of meeting the latency and computational efficiency required to operate on low-power mobile devices, application history prohibiting the use of large-scale networks such as VGG16[14] or variations of the famous ResNet[15]

3. DESCRIPTION OF MODELS

Each model implemented contains a selection of configurable attributes which serve as the tunable hyperparameters available in optimizing the respective model for each task. In general, there are two model methodologies implemented, a bespoke and full-trainable CNN classifier, followed by the application of transfer learning whereby pretrained 'MobileNet' networks are used as generic and interchangeable feature extractors.

3.1 Convolutional Neural Network Classifier

The first model design is a tunable convolutional neural network. The design principle employed in this architecture is the use of three convolutional blocks to enable the neural network to capture both high- and low-level features for each task, as such providing sufficient feature extractor utilities. Within each convolutional block, there is an interchangeable pooling layer that is determined by the tuning process, where this can be either a 'maximum' or 'average' pooling layer. The number of filters for each convolutional layer, also regarded as the number of neurons, doubles that of the previous layer, where the input weights of each neuron form convolutional kernels. A feature map is resulted following the application of each layer of filters, as such a higher-dimensional latent representation of the input for each task is progressively built through the network. A fixed kernel size of (3x3) is used to enable the extraction of sufficiently detailed and localized features for the two datasets considered in this report. This kernel size was deemed optimal through experimentation as ROIs covered in each task typically regarded localized features of facial regions, noting that an odd-dimension size was used to perform more robustly with distorted images, crucial for the data augmentation methods discussed later to be effective. Finally, padding is implemented with the 'same' approach as well as each convolutional layer being activated by the ReLU function, again chosen from experimental analysis that signified their appropriateness for image classification task.

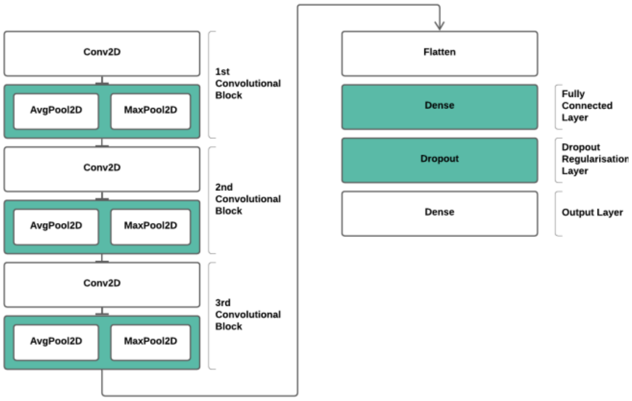


Figure 1: Tunable CNN Architecture (Green blocks indicate the tunable layers)

Following this, the convolutional blocks feed the latent image data through a flattening layer to transform the pooled feature map into a single column for use in the fully connected layer. Fully connected layers are generally seen as a 'cheap' technique of learning non-linear combinations of features received from prior convolutional blocks before moving directly to the final output layer. in this dense layer, a tunable number of neurons is implemented to optimize the configuration for the network and make it less sensitive to the weights of specific neurons. As this can be highly dependent on a given different datasets. Finally, a dropout layer is used between the fully connected layer and the output layer as a form of regularization that helps to prevent overfitting. This technique involves randomly ignored certain neurons in the training process to improve the robustness of dataset, the dropout proportion is automatically tuned for the use case. Only one dropout layer was used given the relatively low network complexity that would make overfitting less of a consequence of the architecture, and more the nature of the data. The culmination of these layers atop the convolutional feature extractor forms the image classification features of the model, whereby the final output layer implements several dense units that is mapped to the number of classes for the given task. Here, the activation function is adaptively chosen to be best-suited to the type of classification task: for multi-class challenges, the SoftMax activation function is used, and for binary tasks, the sigmoid activation function is used.

3.2 - Fine-Tuned Classifier

The second model design leverages interchange pre-trained 'base' models, fine-tuned as feature extractors and connected a simple feed forward neural network regarded as the 'head' model. Inspired by the potential impact of mobile-oriented neural networks, the interchangeable base models are each considered to be 'MobileNets' with comparable depth, size, and number of parameters. In particular, the selected architetcure are: MobileNet [1], MobileNetV2[6] and NASNetMobile[7]. Each base model is harnessed and configured in the same manner, each of which has all layers 'frozen' except for the final convolutional block, using a fixed input size of 224x224. Each of these base models is initialized with weights pre-trained on ImageNet data. Instantiating these based models alongside ImageNet weights provides a means of boosting performance in training neural networks on small datasets. ImageNet provides a wealth of richly diversified image datasets of more than 14 million images with classes highly related to the tasks in this report. As such, the pretrained models harnessing ImageNet weights provides a more robust starting point for training neural networks to classify new images as the use of a broader image set makes the network capable of capturing generic features well, whereby this technique is expected to lead to a lower generalization error as the model has been exposed to a wide variety of data ahead of time.

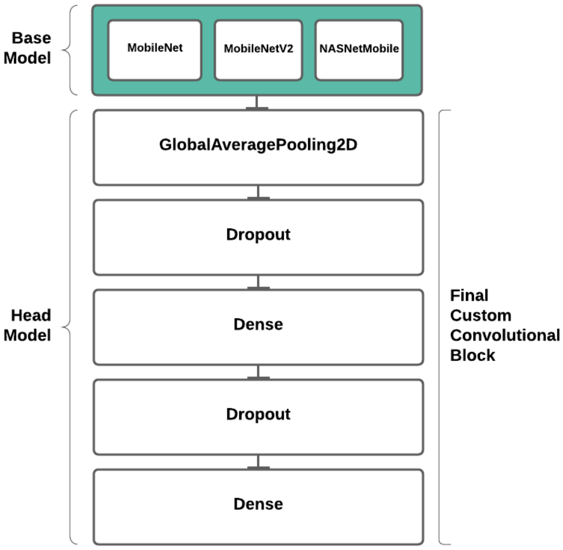


Figure 2: Fine-tined Classifier using interchangeable pretrained 'MobileNets' to perform the feature extraction process (Green blocks indicate the tunable layers)

Firstly, the base model output is fed into a global average pooling layer before reaching the fully connected layer. Here, the sufficiently detailed feature maps provided by the convolution blocks are converted into singular averaged values that help the network to recognize the presence of a particular class. This intermediary layer has been experimentally shown to routinely improve model performance due to the 'naiveness' of the classifier head model from the feature-extracting base model, ultimately helping to reduce overfitting as no new parameters need to be learned in this layer. Following this, an additional tunable dropout regularization layer is inserted to help manage the added likelihood of overfitting when using models with greater complexity. Finally, the fully connected layer is reached in the same sequence as that found in the prior convolutional neural network model, where the model is ultimately terminated with a dense layer containing a number of connect neurons matching the number of desired output classes.

3.3 – Common Components

Both models also have overarching parameters that are available to tune in all instances. Firstly, the models are trained with a tunable learning rate as this has been shown to help manage overfitting, whereby this is progressively decreased on the occurrence of 'plateaus'. Secondly, a suitable loss function is crucially chosen for the suitability of the given task, namely this is interchanged between both binary and categorical cross-entropy for binary and multi-class classification tasks respectively. Thirdly, three different optimization algorithms are leveraged in the compilation of a given model permutation: Stochastic Gradient Descent, RMSporp and Adam.

4. IMPLEMENTATION PIPELINE

4.1 – Data Splitting

The final data preparation activity was to split the original dataset into three purposeful subsets: training, validation, and testing. When training machine learning models, a common challenge that is faced is that of 'overfitting'. Here, a given model is fit well to training data, but is not capable of making similarly accurate predictions on unforeseen data, and thus is said to be unable to 'generalize'. The act of splitting data is to perform a technique known as cross-validation, coordinated subsets to serve different purposes, for training or performance evaluation. The validation subset is exposed at the end of each training epoch to contrast the accuracy performance achieved when classifying training data with when the model is exposed to new data and, often providing a good indicated of the general model quality. Once the given model is fully trained, evaluation of the test set can commence, providing the final verdict on its overall model performance. Each task implements a 60 / 20 /20 data split for training, validation, and testing data respectively.

4.2 – Region of Interest Detection & Extraction

An effective step in preparing image datasets for further processing in classification activities is to detect and extract the region-of-interest (ROI). The first step is to detect the existence of the ROI, and if successfully found, proceeding to remove the unwanted remaining regions in the image. In each task, the ROI extraction method is chosen to best meet the needs of the task, which in this report include simple coordinate-based image crops were used to extract the ROI, pre-trained Haar Cascade classifies trained on the specific desired facial feature, or the more sophisticated HOG detector implementing the 68-pint facial landmark location targets

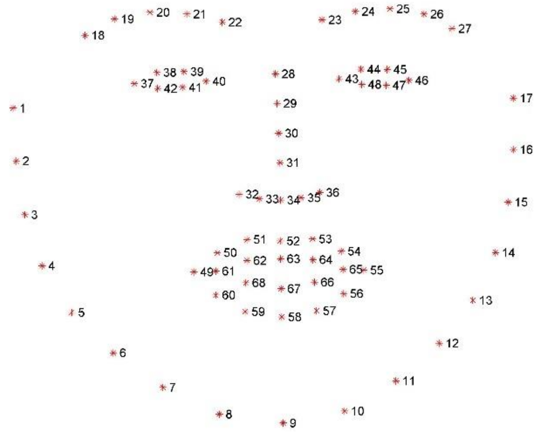


Figure 3: 68-point landmark locations used in HOG ROI extraction processes [8]

In any case, if the desired Roi could not be detected, the given image would be removed from the dataset to mitigate the detrimental influence of low-quality data.

4.3 – Image Augmentation

Image data augmentation is a technique used to artificially expand the size of the training data by producing manipulated variations of the underlying images. Augmentation methodologies have been shown increase the generalizability as this subjects models to more diverse range of relevant images that might be observed in real-world settings. The types of transformations used in this report include shifting, rotating, zooming, and flipping.

An additional measure that has been shown to boost performance is that of 'centering' the image data, a form of data normalization. This achieved by performing a sample-wise subtraction of the mean pixel value of a given batch of images in the preprocessing stages of each task, followed by the subsequent division by the standard deviation of pixel values to normalize each image feature to a z-score quantity. The justification for this process is linked to the nature of how neural network operate, whereby in the training process, many consecutive matrix multiplications and biases are applied to the initial input to stimulate activations that are then backpropagated with gradients to train the model. To constrain the scale of the gradients that may be generated, the given values are normalized to a standard range. In general, this has been shown to enable neural networks to learn faster as gradient values act uniformly for each color channel.

4.2 – Hyper Parameter Tuning

Hyperparameter tuning is a fundamental necessity in the development of performant neural networks. There are a broad range of optimization algorithms that enable the testing of alternating parameter combinations to experimentally converge to an optimal design. In performing the parameter search, the HyperBand optimization algorithm was implemented in each task as experimentation highlighted superior yield in trials per minute, in line with the intended focus of this framework on boosting performing through adaptive resource allocation methods [16].

4.4 - Training

Once the tuning process has completed for each available model variation, the next step is to train these models with the prepared training and validation data subsets. Training involves the process of 'fitting' a model on a set of input data to produce a desired output, in this case of this report, a correctly classified image. In all cases, validation accuracy is the performance metric that is intended to be maximized. In configuring the training process, a series of sane defaults were used for each model in response to insights derived from experimental testing. For example, it was routinely observed that models would reach peak validation accuracy or overfit within 10 epochs. As such, this was set to be a fixed amount. As seen in the hyperparameter tuning step, automated processes are also employed in the training process manage the model training life cycle in a more manageable and scalable manner. Namely, these include checkpointing, exporting the current model weights to aid in long-running sessions, early-stopping used to terminate the training process if the desired metric reaches its peak value with no further improvement, and adaptive learning-rate reduction, used to attempt to dynamically reduce this parameter if the desired metric also stagnates as this has been shown to boost performance.

4.5 – Software Libraries

The implementation of image classification pipelines discussed in this report involved extensive use of popular machine learning libraries and tooling. The most crucial libraries were TensorFlow, a lower-level machine learning 'backend' engine performing the fundamental matrix tensor communications, as well as Keras, a highly abstracted deep learning library used to define and architect neural networks. Furthermore, the cv2 and dlib packages were used in the preprocessing steps and ROI detection methods respectively. A full list of packages used can be found in the project repository.

5. BINARY-CLASS CLASSIFICATION TASKS

The CelebA dataset was used in the implementation of the binary image classification tasks. This dataset contained 5,000 profile images of famous celebrities with a notable degree of variability in regards to the orientation and positioning of the subject in a sample image, as well as the occurrence of frequent noise in the background setting of the profile subject. The labels associated with this dataset include gender and smiling classification, whereby each of these is explore in the subsequent subtasks. Importantly, exploratory data analysis had shown that there was a near-equal presence of each class, as well as fixed sized images of 178 x 218 pixels. As well as this, all images were encoded with three channels (RGB) and consistently stored in the .jpg format.

5.1 Task A1: Gender Classification

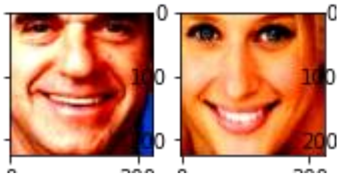
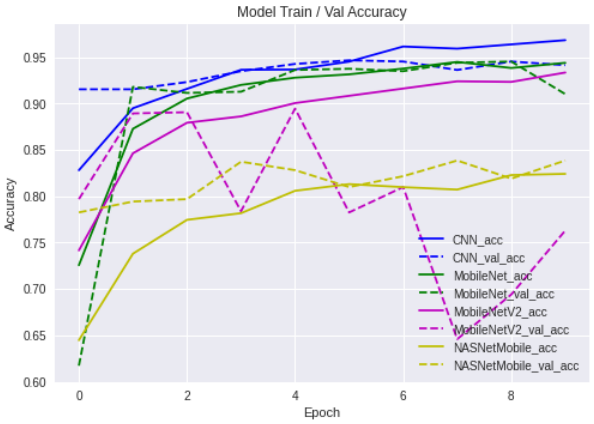


Figure 3: Resulting images following preprocessing and augmentation steps in task A1

The first binary classification task involves gender detection for the CelebA dataset. Here, the dataset was preprocessed to organize images into their respective classes and proceeded to perform region of interest (ROI) detection and extraction. The aforementioned HOG algorithm was used to firstly detect the presence of a face, trained with the 68 point landmark detector, where each of the 68 points was specified for use in the cropping of the facial region, this is because unlike the other tasks explored in this report, no single facial feature provides sufficient information to a neural network to learn about gender, and this was experimentally released by aggressively cropping the face, nose and mouth regions to discover if facial hair or makeup would be more insightful, but such models routinely underperformed as compared to the full facial crop.



(Above) Task A1 learning curves for each model implemented

This step was a necessary data cleaning exercise that improves the quality of the available dataset by exposing the eventual neural networks to only the important features ahead of time, as well as reducing the potential background noise that may be present. Because of this step, 3.6% of the images were immediately removed from the available dataset as the ROI could not be detected, indicating a highly disruptive profile orientation that would be useful to extract features from, anomalies if you will. As well as this, image samples were augmented to introduce random rotation, horizontal, sheer and shift transformations to expose the models to variations likely to occur within the dataset or other highly related real-world images of human facial profiles. As well as this, each image was normalized through sample wise mean pixel value subtraction and division by the standard deviation of these pixel values found in the sample data.

5.2 Task A2: Smile Detection

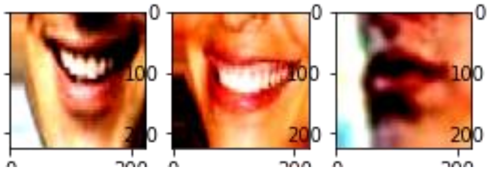


Figure 4: Resulting images following preprocessing and augmentation steps in task A2

The second binary classification task involved the classification whether as subject was smiling for a given sample image of the CelebA dataset. The implementation of this task also begins with vital preprocessing steps to clean the data and reduce the present noise when possible ahead of time. Similar to task A1, here we implemented the HOG facial to firstly detect the mouth region within a profile image, and if detected, subsequently proceeding to crop this region using points 48 to 67. This process equally led to a reduction of the available dataset size by 3.6%, likely due to the same challenge of unpredictable orientation and position of the front profile of the subject, where the mouth is a major feature. Supplying the neural networks with cropped mouth regions for training activities would enable a more precise understanding of the ROI with the ability to understanding the differences in mouths that are smiling and those that are not. As is common with many image classification problems, local color intensity and edge features will be features learned by the convolutional-based neural networks identify, for example, the presence of white teeth immediately surrounded by a dark lip region, likely signifying a subject that is smiling. Following this, image samples were normalized with sample-wise mean pixel value subtraction and division by the standard deviation value, as well as randomly transformed in augmentation steps to enable the model to learn the unpredictable variation of orientation and positioning of the mouth region in real world images of human faces, thus empowering the model to be more generalizable to unseen datasets of human faces.

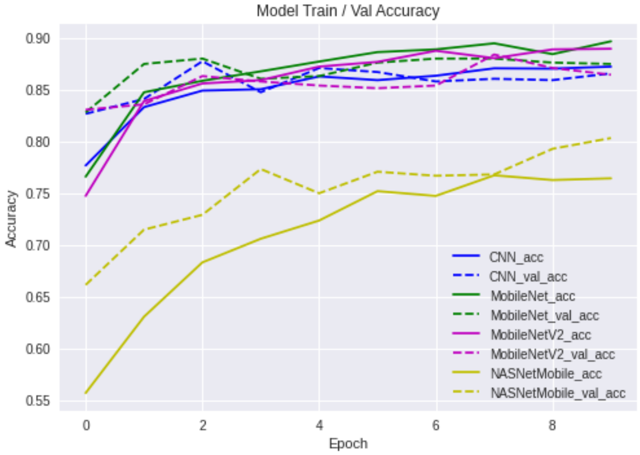


Figure 5: Task A2 learning curves for each model implemented

6. MULTI-CLASS CLASSIFICATION TASKS

The CartoonSet dataset was used in the implementation of the multiclass image classification tasks. This dataset contains 10,000 profile images of randomly generated cartoon faces, each with interchangeable features. The labels provided for this dataset included 5-class groups associated with each of the subtasks in this section, namely the face shape and the eye-color features. Each of these class types had a near equal presence in the dataset thus minimizing the overfitting risks caused by imbalanced datasets. Importantly, exploratory data analysis had shown that there was no rotational or positional variance in image samples, as well as the images being a fixed size of 178 x 218 pixels. All images encoded with three channels (RGB) and consistently stored in the .png format.

6.1 – Task B1: Face Shape Classification

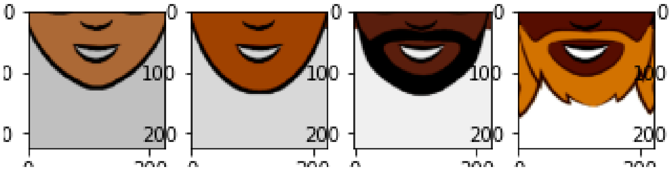


Figure 6: Resulting images following preprocessing and augmentation steps in task B1

The first task involved classifying the different face shapes found in the each of the cartoon avatar variations. The first implementation step for this task was to identify a suitable region of interest (ROI) to extract the most valuable information to feed into the neural networks. This process concluded with selecting the jawline area, from the chin to the ears, as an optimal region that best depicted the different face shape classes, as well as providing an opportunity to remove noise impacts of head hair variations. Although, this still left the impact of facial hair covering the target area. this was deemed to be a reasonable tradeoff. This process was executed using simple coordinate-based cropping due to the unpredictable nature of relying on the HOG facial point detector. Here, the intended approach was to use points 3 to 15 in the 68-point landmark locations to capture and crop the same jawline region as is extracted with simple cropping, however the recurrent impact of facial hair in many of the samples resulted into poor and variable crops that negatively impacted the performance of the models and incurred a great computational cost in performing the extraction. Simple cropping outperformed the more sophisticated approach. Following ROI extraction process, the image samples were normalized by subtracting the sample-wise mean pixel value from the image data and divided by the standard deviation.

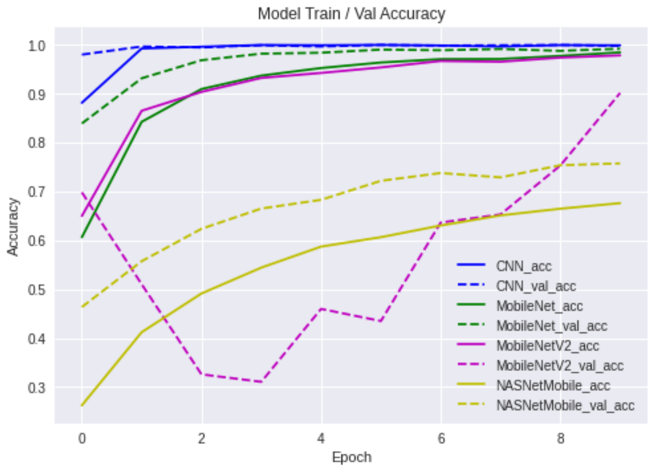
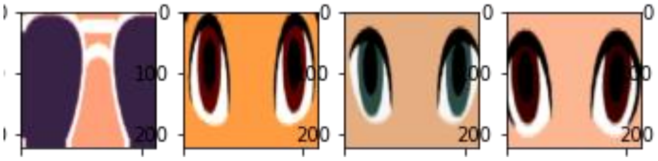


Figure 7: Task B1 learning curves for each model implemented

Note, in this task the implementation of data augmentation transforms was revoked. Interestingly, every model yielded superior performance when training on images that had not been randomly transformed. This is clearly due to the consistent positioning and repeatable nature of images in the Cartoon Set dataset. In this case, the injection of artificially diversified data resulted in too much noise and variability that the neural network is less successful at learning the basis of the true underlying data, and as such performance is seen to diminish. Clearly data augmentation practices cannot be treated as a consistent solution, and one must always suit the preprocessing approach to the nature of the underlying dataset

6.2 – Task B2: Eye Color Classification



(above) Figure 8: Resulting images following preprocessing and augmentation steps in task B2

The second multiclass classification task involved the classification of the different eye colors. The first step here was to perform sufficient ROI extraction to ensure that each neural network model was provided with information that was relevant to the task. Exploratory data analysis had highlighted that a considerable portion of the image samples contain subjects that are wearing sunglasses such that the eye-color was either partially or completely blocked. Given that the number of samples wearing sunglasses represented approximately 20% of the available data, it was deemed optimal to expand the classes available in this task to include an additional 'sunglasses' label. This would help mitigate the detrimental impact of being a model being confronted with a subject wearing sunglasses that would likely lead to inaccurate class labelling attribution. To perform this ROI process, a Haar Cascade [9] trained to classify human eyes was utilized to detect the occurrence of eyes, where the lack thereof likely implied that a given sample was wearing sunglasses, whereby in both cases a consistent coordinate location was cropped regardless if eyes could be detected or not. Following this, each image was normalized by subtracting the sample-wise mean pixel value in each batch of images and subsequently dividing by the standard deviation.

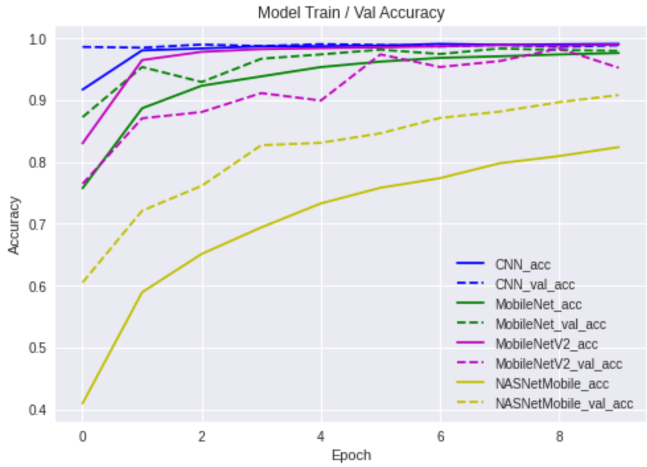


Figure 9: Task B2 learning curves for each model implemented

As was observed in task B1, the process of augmenting these images was shown to be detrimental to the performance of each neural network trained on such data, adding unnecessary complexity. Instead, the detection of an eye region and subsequent cropping provided the models with information that enabled the simpler preprocessing step to surpass that of the more sophisticated data augmentation, whereby because the eye region was fixed in each image, the models could not benefit from artificial exposure to potential image variations that it will not face with unseen dat from this dataset. If such a model was to be trained on images of human eyes or a diverse range of cartoon characters, the models may indeed benefit from more sophisticated pre-possessing, but in this case this was not observed,

6. EXPERIMENTAL RESULTS AND ANALYSIS

Task	Model	Train Acc	Val Acc	Test Acc
A1	CNN	96.2%	94.8%	92.5%
A2	MNet	89.9%	87.2%	83.4%
B1	CNN	99.8%	98.9%	95.4%
B2	CNN	99.9%	99.8%	96.8%

(Mnet = MobileNet)
Table specifying the best training, validation testing accuracy achieved by any of the separate models used in each task.

Despite the variations in the challenges presented in this report and the diverse nature of each respective dataset, there were a series of recurring insights. The convolutions neural network architecture used throughout the implementation of each task appeared to perform most consistently despite its relative simplicity compared to the other approaches. This was undoubtedly a experimental surprise, as it was expected that more complex, pretrained models would always outperform bespoke model developed locally.

As with all models, fine-tuning to the underlying datasets is a fundamental necessity for any well-performing model. As such, for the dimensionality scale of both the CelebA and Cartoon Set datasets, the CNN neural network design appeared to provide sufficient depth, in terms of the number of convolutional block feature extractors, to construct a sufficiently detailed feature map of each latent image that enabled the full-connected classification layers to identify the present non-linear relationships more successfully. Furthermore, the limited discrepancy between the training and validation accuracy indicates that such a model would perform well on unseen data of a similar nature, as shown by the comparable test accuracy score.

In general, given then resulting models produced from the hyperparameter tuning process implemented for each task it appeared that the fine-tuned models with greater complexity suffered learning undesirable features from the input data, thus resulting in overfitting, as illustrated by the more significant training/validation accuracy discrepancies. This was particularly true for fine tuned models using the MobileNetV2 base model architecture. As this architecture is designed for speed improvements upon its predecessor, it is believed that the in-training adaptive learning rate features negatively impacted the learning process of these models, whereby this architecture appears to be sensitive to the suite of optimization configurations implemented. In contrast, the NASNetMobile architecture routinely unfit to data provided in each task. This signifies the model’s inability to effectively capture the underlying desired features from the input data provided and indicates that such a model would perform poorly on subsequent unseen data.
Please refer to the appendices for the complete collection of data plots for each task.

7. CONCLUSION

This report covered the application of a selection of deep neural network models in performing image classification, focusing in particular in the comparison of streamlined convolutional neural networks with the promising advent of 'MobileNets' [1].

A consistent implementation was used in each task to address the fundamental aspects of image classification activities, in particular the process of noise reduction through region of interest (ROI) detection and extraction proved vital in boosting the performance of each neural network permutation and ultimately optimizing a given neural network architecture to the task at hand.

Despite the higher-level architecture, convolutional neural networks were shown to be highly robust and versatile to several image-based classification challenges, validating popularity they have received for use in computer vision applications in recent times.

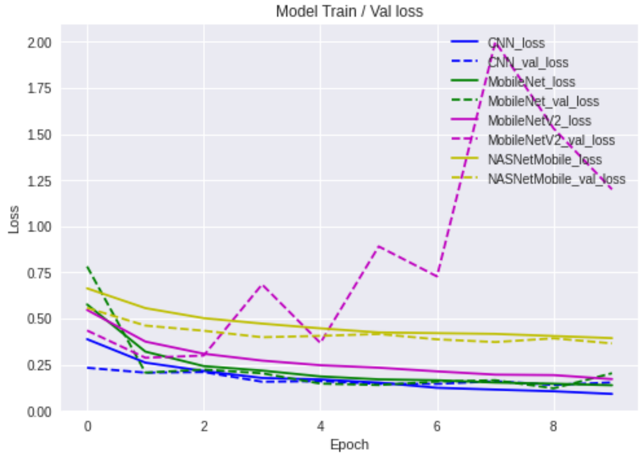
Just as data augmentation was seen to negatively impact the overall network performance, so too was the empirically observed contribution of applying transfer learning techniques with larger networks pretrained on more diversified data, whereby it is believed that the increased sophistication introduced to help mitigate the impact of under of overfitting may consequently have a detrimental effect on the overall model as too many features are learned from the underlying data making it more difficult to accurately predict the desired labels on unseen data. However, because the models developed implement automated tuning for new underlying datasets, it may well be observed that their overall performance 'leaderboard' may be rearranged with sufficient tuning trials, unfortunately this is a prohibiting possibility with limited computing resources.

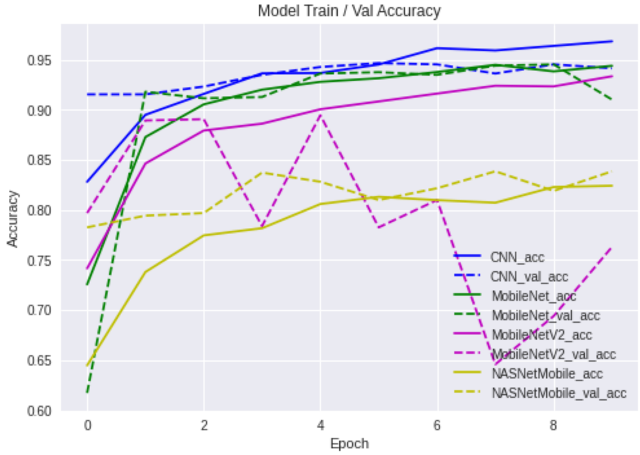
There are numerous further improvements that one might make in pursuit of the overarching goals of this report and determining more insightful outcomes. For example, other than increasing the dataset size which most deep learning models will benefit from in terms of generalizability, a more comprehensive hyperparameter search space could be used to not only tweak the inner workings of layers and their subcomponents during the training process, but to also tune the infrastructure of the training process itself, such as the tuner trials, the epochs, the batch size etc. This will undoubtedly uncover higher fidelity insights about the optimal combination of configurable parameters and utility converge to higher performing models.

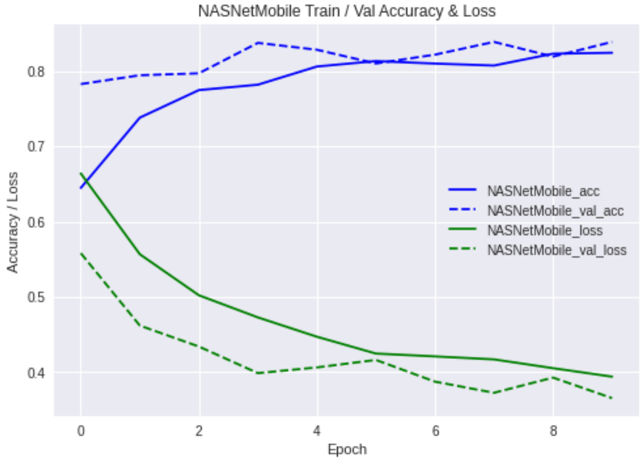
APPENDICES

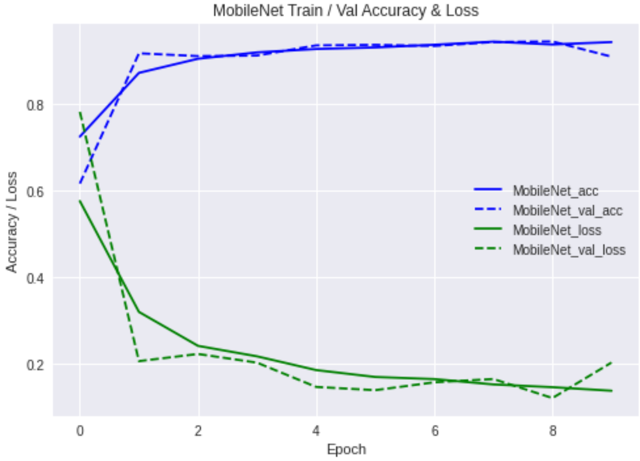
TASK A1 PLOTS

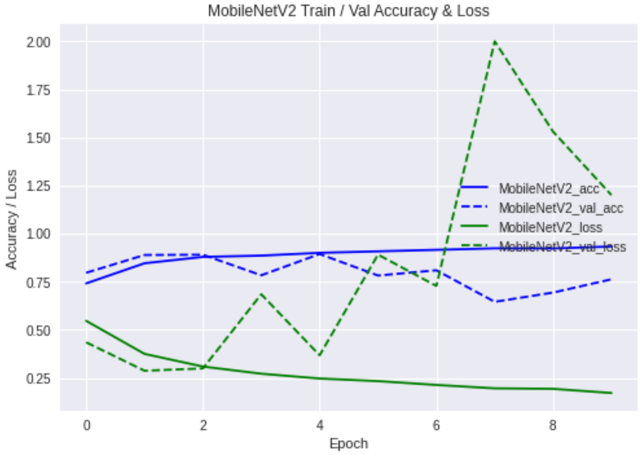
Overall plots as well as plots for each individual model.

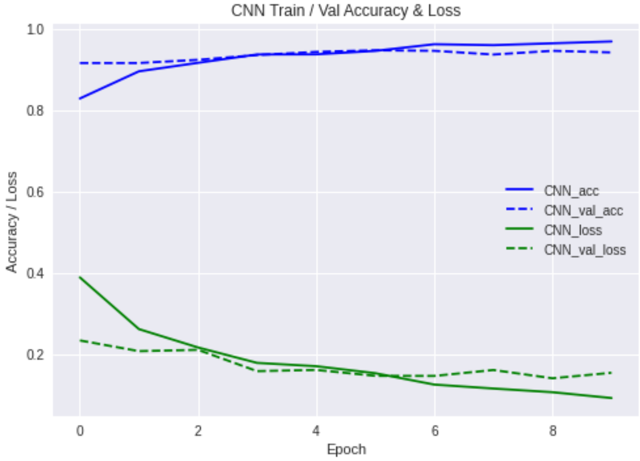






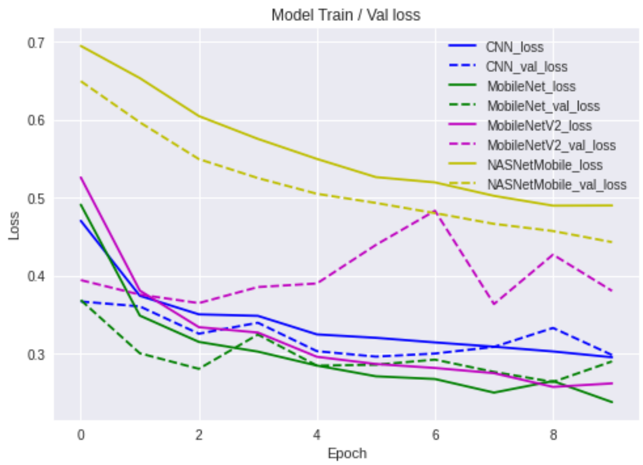


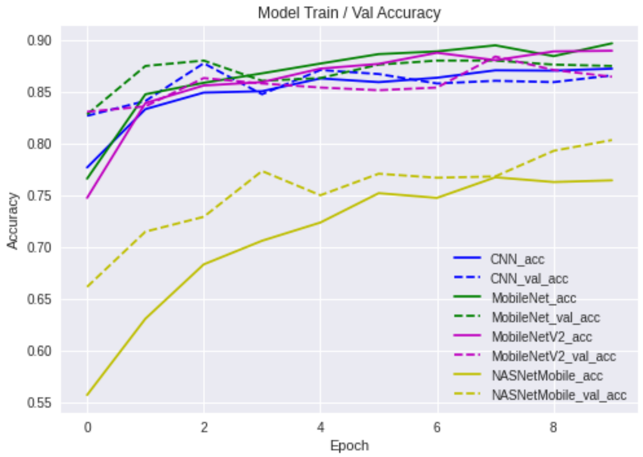


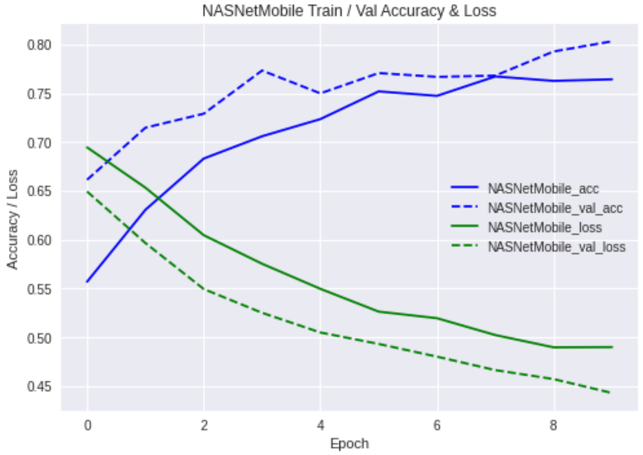


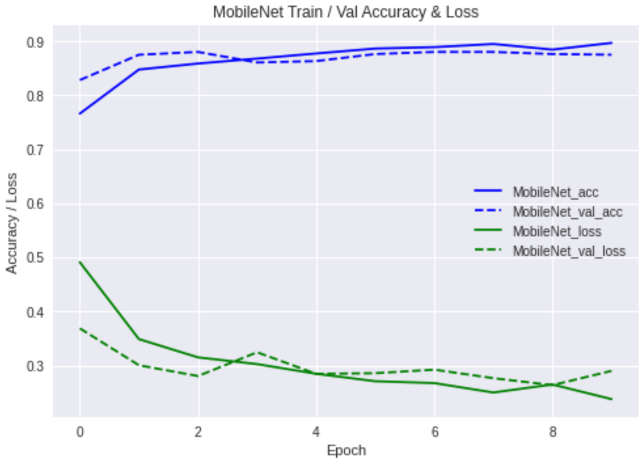
TASK A2 PLOTS

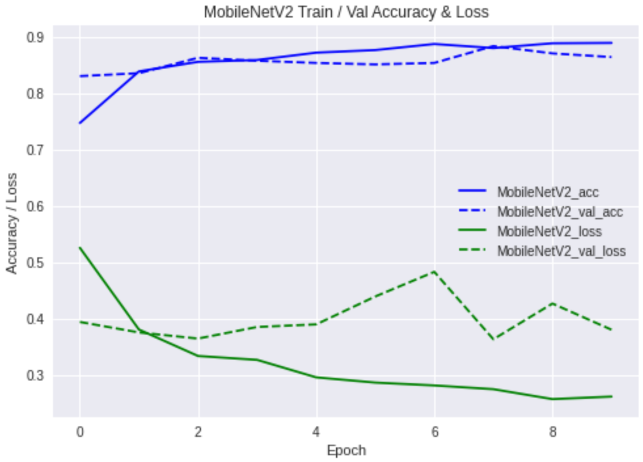
Overall plots as well as plots for each individual model.

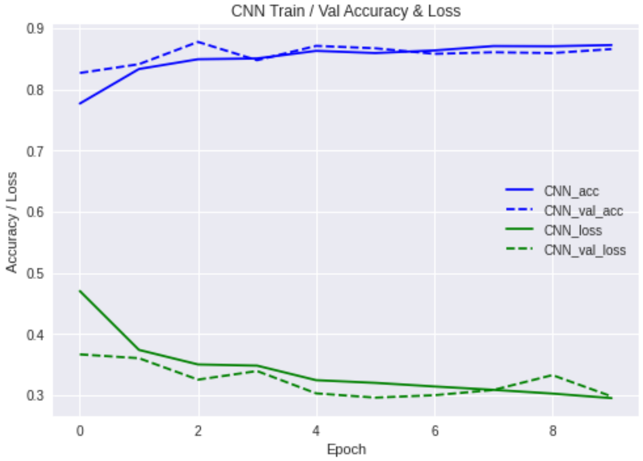








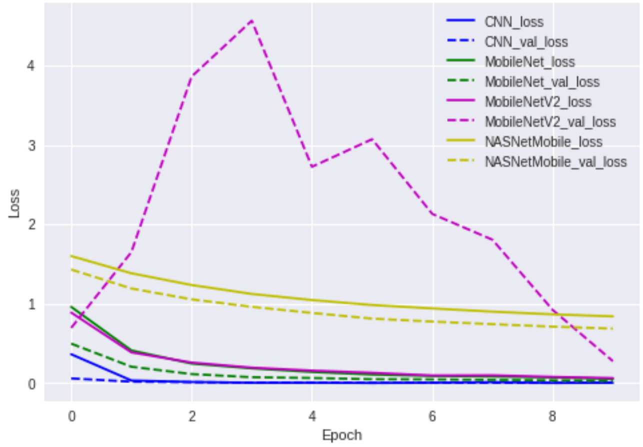




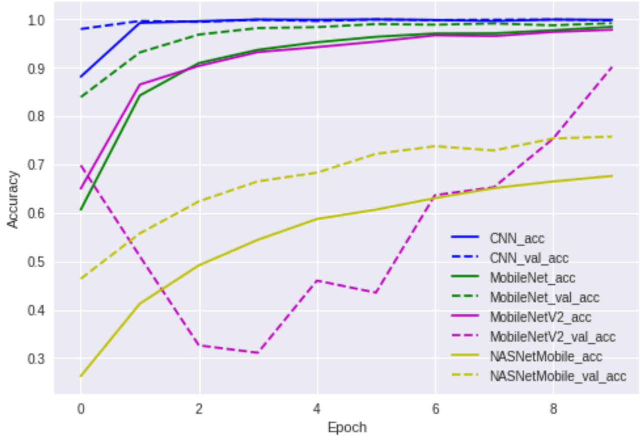
TASK B1 PLOT

Overall plots as well as plots for each individual model.

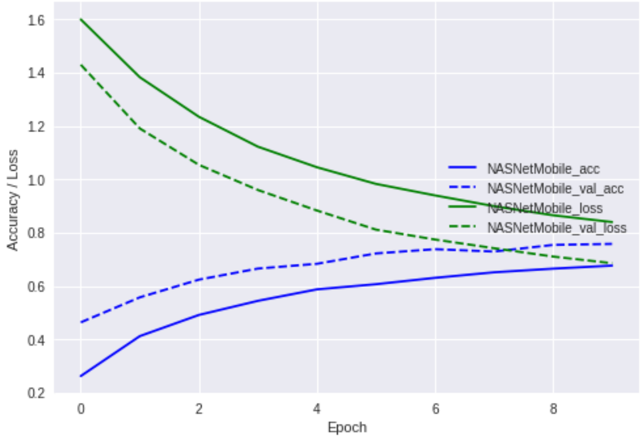
Model Train / Val loss



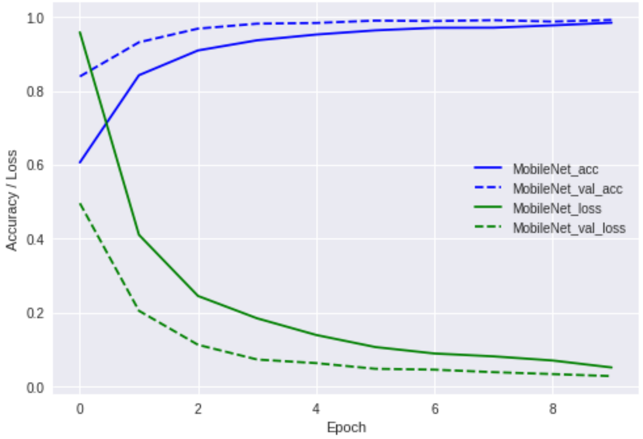
Model Train / Val Accuracy



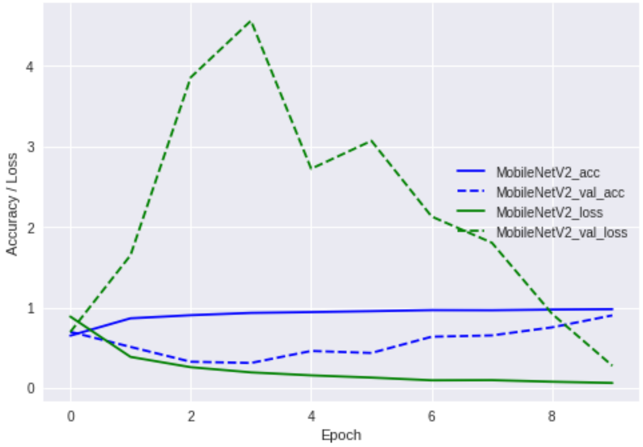
NASNetMobile Train / Val Accuracy & Loss



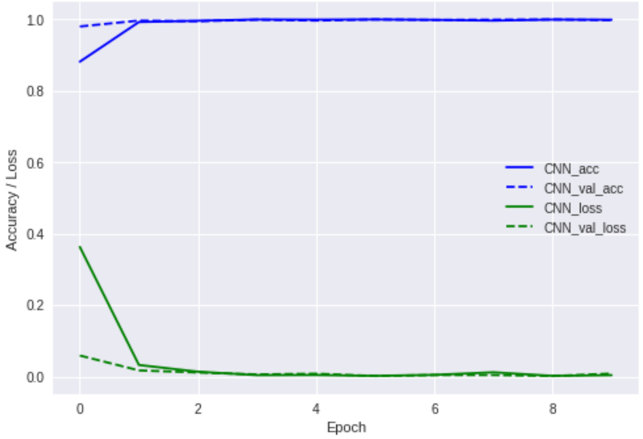
MobileNet Train / Val Accuracy & Loss



MobileNetV2 Train / Val Accuracy & Loss



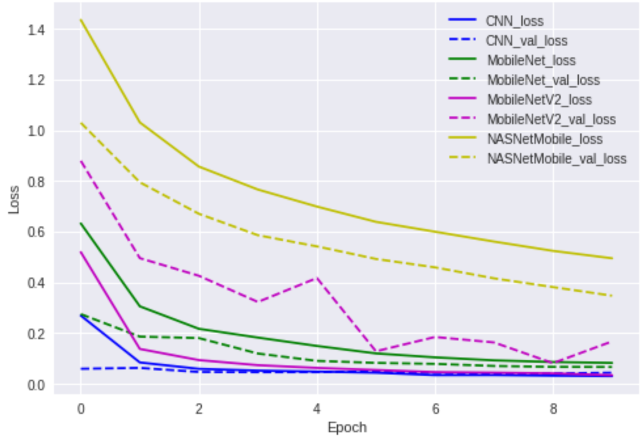
CNN Train / Val Accuracy & Loss



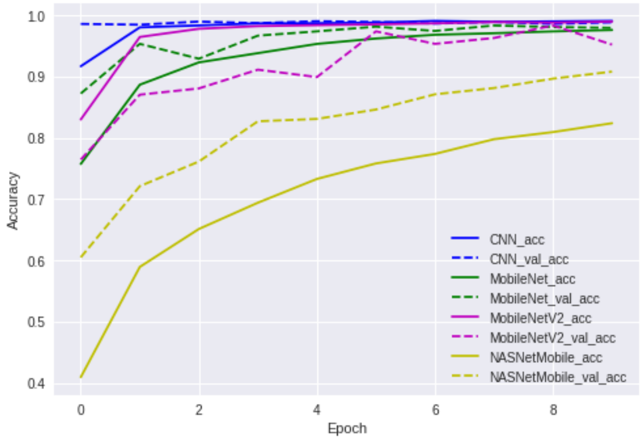
TASK B2 PLOTS

Overall plots as well as plots for each individual model.

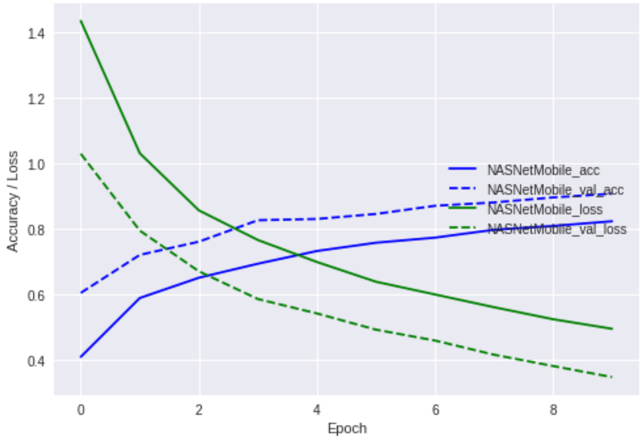
Model Train / Val loss



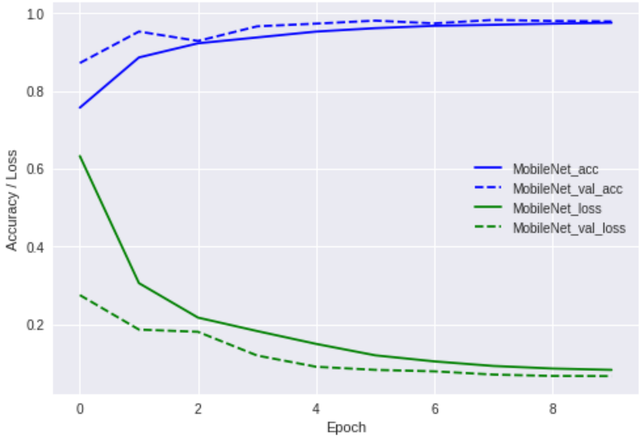
Model Train / Val Accuracy



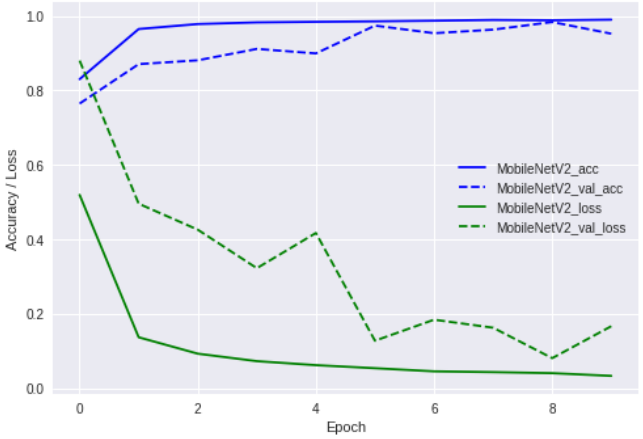
NASNetMobile Train / Val Accuracy & Loss



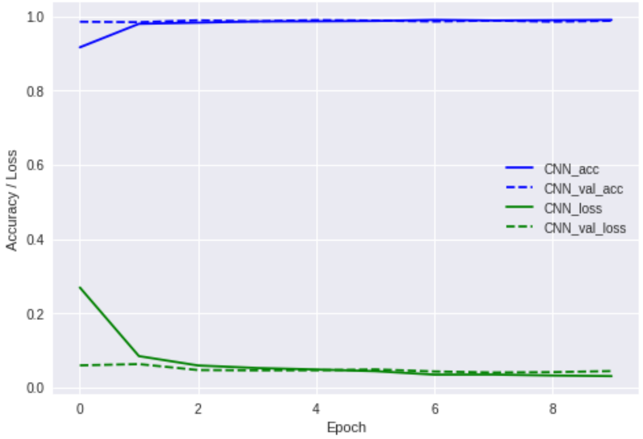
MobileNet Train / Val Accuracy & Loss



MobileNetV2 Train / Val Accuracy & Loss



CNN Train / Val Accuracy & Loss



REFERENCES

[1] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam

[2] Histograms of oriented gradients for human detection, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)

[3] Rapid Object Detection using a Boosted Cascade of Simple Features, Paul Viola Michael Jones

[4] A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects, Zewen Li, Wenjie Yang, Shouheng Peng, Fan Liu

[5] Transfer Learning for Illustration Classification, Manuel Lagunas1 Elena Garces

[6] MobileNetV2: Inverted Residuals and Linear Bottlenecks, Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen

[7] Learning Transferable Architectures for Scalable Image Recognition, Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le

[8] 68-point HOG detector, https://github.com/davisking/dlib-models/blob/master/shape_predictor_68_face_landmarks.dat.bz2

[9] Haar cascade eye classifier, https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_eye.xml

[10] TensorFlow Python Library, https://www.tensorflow.org/

[11] Keras Python Library, https://keras.io/

[12] dlib library [13], http://dlib.net/

[13] OpenCV library, https://opencv.org/

[14] Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan, Andrew Zisserman

[15] Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

[16] Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization, Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, Ameet Talwalkar