# Image CAPTCHA Handover Documents

Prepared by Tan Yoon Lok & Jack Lenigas

**Project Summary**

The project's objective is aimed to examined and created a unique model with that helps in improving the performance of a model in cracking image CAPTCHA, aside from using the existing solution.

**Project Plan**

In this trimester, we aimed to create a stacked model utilizing deep learning techniques (conventional neural network) in classifying the images. In order to ensure the objective work well, a publicly available dataset- CIFAR-10 (consist of 60,000 images) are being used to build the model, supporting by the article named 'Automating the Bypass of Image- based CAPTCHA and Accessing Security'. One of the reasons CIFAR-10 is used is because of the variety of images. The data consist of 10 different classes of images, that included: Airplane, Automobile, Bird, Cat, Dog, Deer, Frog, Horse, Ship and Truck.

**What is Image Classification?**

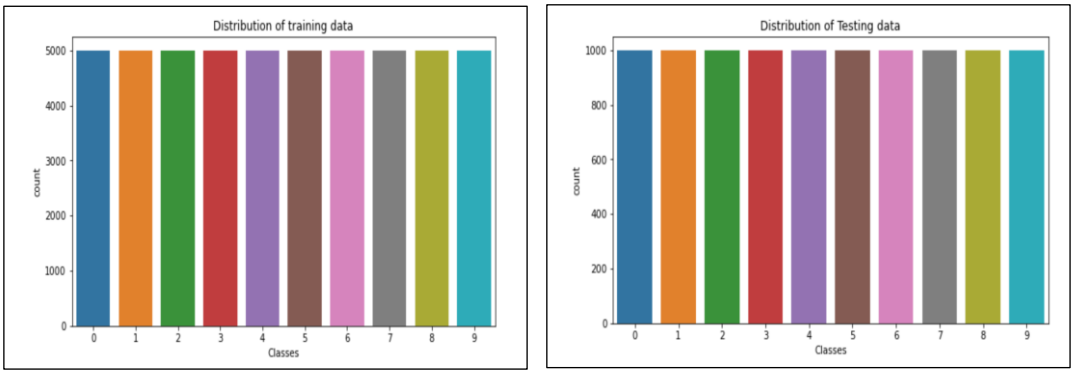
Image classification is a supervised learning problem in which a model is trained to recognize a collection of target classes (objects to identify in pictures) using labelled sample photos. Picture classification, localization, image segmentation, and object identification are some of the key challenges in computer vision. Image classification is one of the most essential issues among them. Image classification applications are employed in a variety of applications, including medical imaging, satellite image object identification, traffic management systems, and, in the case of our study, cracking Image CAPTCHAs. A basic summary of how it works is that a computer analyses an image in the form of its pixels. It accomplishes this by treating the picture as an array of matrices, the size of which is determined by the image resolution. Image classification is accomplished in digital image processing by automatically grouping these sets of pixels into defined categories, referred to as "classes". Traditional machine learning and AI-based Deep Learning are the two major methodologies of image classification. Deep learning will be the major focus of this paper since it offers the most versatility in terms of models that may be investigated. Convolutional Neural Networks (CNNs) will be investigated in this case of image classification because they are purpose-built for the task. A CNN is a machine learning framework that was created utilizing machine learning ideas. Without the need for human interaction, CNNs can learn and train from data on their own. When employing CNNs, just a little amount of pre-processing is required because they create and adapt their own picture filters, which must be properly designed for most algorithms and models.

**CNN Terminology**

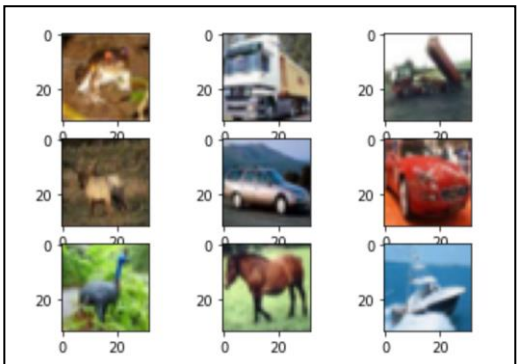
A CNN's top-level overview is as follows: A neuron is the fundamental unit of a CNN; they are statistical functions that calculate the weighted average of inputs and apply a mathematical function to the output. Layers are a collection of neurons, each of which has a specific purpose. Depending on the depth sought and the CNN's function, a CNN system can include anywhere from a few to hundreds of layers.

**Pre-Processing**

Prior building a model, we first checking the data by visualizing the classes of data (both train and test dataset) as well as the sample images to obtain early understand of the image data. With 60,000 images of data, the data is then divided into 2, that is 50,000 images into train dataset and 10,000 images into test dataset.




Examined both training and test dataset



Plotting of Sample Images

After examined a sample of images, it then follows by a normalization of the data before a model is being created. The data is then being normalized via $(x - \mu)/\sigma$.

```
# Normalize the mean and standard deviation
mu= np.mean(x_train)
std= np.std(x_train)

# Normalize the data by using X-mu/SD
x_train= (x_train-mu)/std
x_test= (x_test-mu)/std
```

After that, a conventional neural network model is then being constructed as below:

```
# Model 1 with kernel regularizer, padding, and activation ReLu applied 30 epochs tested without data augmentation
reg= None
num_filters= 32
ac= 'relu'
adm= Adam(learning_rate= 0.001, decay= 0, beta_1= 0.9, beta_2= 0.999, epsilon= 1e-08)
opt= adm
drop_dense= 0.5
drop_conv =0

model = Sequential()

model.add(Conv2D(num_filters, (3, 3), activation= ac, kernel_regularizer= reg, input_shape= (img_rows, img_cols, channels), padding= 'same'))
model.add(BatchNormalization(axis=-1))
model.add(Conv2D(num_filters, (3, 3), activation= ac, kernel_regularizer= reg, padding= 'same'))
model.add(BatchNormalization(axis=-1))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(drop_conv))

model.add(Conv2D(2*num_filters, (3, 3), activation= ac, kernel_regularizer= reg,padding= 'same'))
model.add(BatchNormalization(axis= -1))
model.add(Conv2D(2*num_filters, (3, 3), activation= ac,kernel_regularizer= reg,padding= 'same'))
model.add(BatchNormalization(axis=-1))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(drop_conv))

model.add(Conv2D(4*num_filters, (3, 3), activation= ac,kernel_regularizer= reg,padding= 'same'))
model.add(BatchNormalization(axis= -1))
model.add(Conv2D(4*num_filters, (3, 3), activation= ac,kernel_regularizer= reg,padding= 'same'))
model.add(BatchNormalization(axis =-1))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(drop_conv))

model.add(Flatten())
model.add(Dense(512, activation= ac,kernel_regularizer= reg))
model.add(BatchNormalization())
model.add(Dropout(drop_dense))
model.add(Dense(num_classes, activation= 'softmax'))

model.compile(loss= 'categorical_crossentropy', metrics= ['accuracy'],optimizer= opt)
```

Not limited to that, a data augmentation technique has also been implement as below using imagedatagenerator function in keras with different parameters tuned to obtain the best output. A number of models has been built with different parameters tuned (Please refer to William's document in github).

Here are some techniques which has been considered to be applied in the project, including data deblurring, and data augmentation.

<u>Blur Classification & Deblurring of Images</u>

Blurring is a major cause of image degradation, reducing the quality of an image to protect against malicious attack. Blurs are generally divided into: Gaussian Blur, Motion Blur, Defocus Blur, Average Blur and so ons. To improve the accuracy of the classification model, low resolution of images needed to be processed and it can achieve by applying image denoising (part of image deblurring). Not limited to that, image deblurring are included image segmentation, iris recognition, microscopy, information retrieval, astronomy, video object extraction, and space observation.

Explanations of different blur types are as below:

(a) Gaussian blur: Effects of this blur is created through Gaussian filter following a bell-shaped curve.
(b) Motion blur: Motion blur can be caused by motion- based activities between camera and scene during the exposure time.
(c) Defocus blur: Defocus blur can be reached by applying optical imaging system to blur the image background and pop-out the main object using large aperture lenses.
(d) Average blur: Average blur can be scattered in both vertical and horizontal direction. The average filter is useful to remove this type of blur especially in the case where noise is affecting the whole image.

Data Augmentation

Data Augmentation is a data-space solution to the problem of limited data and is powerful to enhance the sizes and quality of training dataset. One of the well-known applications of image augmentation is in medical data such as MRI and CT, shown in [5] [7], given the limited of data. [4] discussed geometric transformations, kernel filters, colour space augmentation, random erasing, mixing images, adversarial training, feature augmentation, neural style transfer, generative adversarial network (GAN) and meta learning.

Deep Neural Network (DNN) has been long and successful in computer vision task: image classification, image segmentation and object detection thanks to the evolving of Conventional Neural Network (CNN).

Overfitting is a key metric to evaluate the performance model. A good model loss should decline overtime, following with the training rate declining. In contrast, loss of testing set increase where training set is declining is a clear sign that the model is overfitting. One way to identify overfitting is to plot the accuracy and loss at each epoch. Numbers of overfitting solution are listed below:

(a) Dropout

Dropout is a regularization method that zeros out the activation value of randomly selected neurons during the training. This has forced the network to learn more robust features rather than relying on the predictive capability from a small subset of neurons within in the network. Spatial Dropout dropped the whole feature maps rather than a single neuron.

(a) Batch Normalization

Batch Normalization is one of the techniques that normalize the set of activation layer. Batch Normalization work by subtracting the batch mean from activation and divided by batch standard deviation (x- mu/ sd). The normalization along with standardization is a common technique to preprocess the pixel values.

(b) Transfer Learning

Transfer Learning is a technique that utilize pre-trained model on a new problem. Transfer learning work perfectly with big dataset- ImageNet by training the network and the weight as initial weights for the new classification task to improve the learning process and the development of the new target task. Noted that in any circumstances where the performance of new model is declining, it is considered a negative transfer.

- Inductive Transfer Learning
- Unsupervised Transfer Learning
- Transductive Transfer Learning

Data Augmentation is differed from any of techniques above, as it approaches overfitting issue from the root of the problem. Augmentation techniques artificially inflate the training dataset either by oversampling or data wrapping, and it can be found in Le-Net [xx]- pg 6.

Now, let's discover the techniques of data augments based on image manipulations.

(a) Rotation

Rotation augmentation can be done by rotating the image in both left and right between 1 degree to 359 degree. Noted that high rotation degree will reduce the effect, causing the label of the data no longer preserved post- transformation.

(b) Flipping

Flipping augmentation allowing to flip in both vertically and horizontally.

(c) Cropping

Cropping images can be used as a practical processing step for image data when the dataset is mixed with different height and width dimensions data by cropping a central patch of each image. Besides, random cropping can also be used to provide an effect very similar to translations.

To justify the differences between random cropping and translation is that cropping reduce the size of the image, eg: from (256, 256) → (224, 224), whereas translations preserve the spatial dimensions of the image.

(d) Color Space

Digital image data is usually encoded as a tensor of the dimension (height × width × color channels). Performing augmentations in the color channels space is another strategy that is very practical to implement, and it can be as simple as isolating single color such as R, G, and B. Additionally, the RGB values can be easily manipulated with simple matrix operations to increase or decrease the brightness of the image.

(e) Translation

Shifting images left, right, up, or down can be a very useful transformation to avoid positional bias in the data. For example, if all the images in a dataset are centered, which is common in face recognition datasets, this would require the model to be tested on perfectly centered images as well. As the original image is translated in a direction, the remaining space can be flled with either a constant value such as 0 s or 255 s, or it can be flled with random or Gaussian noise. Tis padding preserves the spatial dimensions of the image post-augmentation.

(f) Noise Injection

Noise injection generally begin by injecting a matrix of random values drawn from a Gaussian distribution to the data, that enable CNNs to learn more robust features in the dataset.

(g) Random Erasing

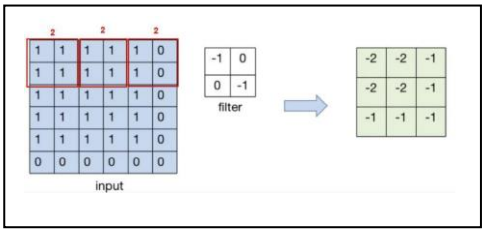
Random easing is another data augmentation technique that interesting to explore, inspired by the mechanism of dropout regularization. This technique designed to combat image recognition issue due to occlusion. Random erasing is useful to combat against occlusion, as it forces model to learn more descriptive features about the image, while preventing it from overfitting to certain visual features in the image. Worth noting, random easing guarantees the network focusing on the entire image rather than just a subset of images.

(h) Kernel filters

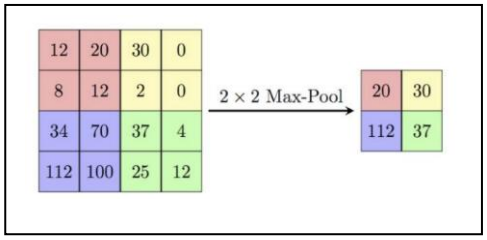
Kernel filters is a very popular image processing technique that sharpen and blur images. The filters work by sliding n x n matrix across an image with a Gaussian blur filter. PatchShuffle

Regularization, a relatively new technique has been proposed, a unique kernel filter that randomly swaps pixel values in an n x n sliding windows.

Conv2D: A kernel (filter) that passes over an image, from left to right, top to bottom, applying a convolution product. The convolution product is an elementwise (or pointwise) multiplication. The sum of this result is the resulting pixel on the output image. Below is a basic example:




MaxPooling2D: A down sampled (pool) feature map is created by calculating the highest value for patches of a feature map and then using that value to generate a down sampled (pool) feature map. After a convolutional layer, it's commonly utilized. It's widely used to quickly decrease the size of an image.




Note that kernel filters are a relative unexplored area for data augmentation, we strongly suggest the future members to investigate kernel filters technique and to apply it into CNN model.

Data Augmentation Based on Deep Learning

When come into feature space augmentation, SMOTE can be very useful in managing issue of imbalance class/ dataset. SMOTE stand for Synthetic Minority Oversampling Technique and it is to apply to the features space by joining the k nearest neighbours to form a new instance in countering imbalance data.

Imbalance data is the data where observed frequencies are very different cross the different possible values of categorical variables. To counteract imbalance data, we can either choose to perform undersampling or oversampling. These techniques do not come without weaknesses.

Weakness of Undersampling: Possibly loss of valuable data

Weakness of Oversampling: Creation of artificial data, which potentially introduce false information to the model.

In contrast to undersampling and oversampling, SMOTE is an improved alternative of oversampling to be applied as it generates synthetic data points rather than duplicated the data point, that potentially impact the model performance. As an output, SMOTE increase recall at the cost of precision.

Noise addition, interpolating and extrapolating are also common forms of feature space augmentation.

Not limited to that, autoencoder is also worth mentioning. The use of autoencoder is particular useful when it comes to feature space augmentations.

In any circumstance where unsupervised deep learning is needed, autoencoder can be useful in dimensional reduction. Autoencoder is a data compression algorithm where the process of compression and decompression are data- specific, lossy, learned automatically from example that can learn to map input to the output data.

Autoencoders are separated into 5: Undercomplete autoencoders, regularized autoencoders, sparse autoencoders, denoising autoencoders, variational autoencoders. In our case, denoising autoencoder can be useful to be implemented to show cleaned output. The example is as below:

Not limited to that, autoencoder is practically useful to perform feature space augmentation. Autoencoders work by having one half of the network, the encoder, map images into low-dimensional vector representations such that the other half of the network, the decoder, can reconstruct these vectors back into the original image. This encoded representation is used for feature space augmentation. Feature space augmentations can be implemented with auto-encoders if it is necessary to reconstruct the new instances back into input space. It is also possible to do feature space augmentation solely by isolating vector representations from a CNN. This is done by cutting of the output layer of the network, such that the output is a low-dimensional vector rather than a class label. Vector representations are then found by training a CNN and then passing the training set through the truncated CNN. Despite the powerful of feature space augmentation, the interpretation of vector data can be its weakness and difficult to achieved.

Besides, numbers of packages of data augmentation have been listed below. We encourage future members to explore different packages to understand the potential and parameters from different packages.

**Table 2.** Common software libraries and frameworks for deep learning image data augmentation.

| Ref. | Name | Geometric | Intensity | Deformable |
|---|---|---|---|---|
| 189 | ImgAug | Y | Y | Y |
| 190 | Augmentor | Y | N | Y |
| 191 | Keras ImageDataGenerator | Y | Y | N |
| 192 | PyTorch Transforms | Y | Y | N |
| 193 | TorchIO | Y | Y | Y |
| 194 | Albumentations | Y | Y | Y |
| 195 | CLoDSA | Y | Y | Y |
| 196 | MONAI | Y | Y | Y |

Faster RCNN

Not limited to CNN, Faster- RCNN has also been explored in this trimester (Please refer JackLenigas, Faster RCNN file). To understand why this is an optimal solution for fast and timely CAPTCHA solving, it is beneficial to understand how R-CNNs work first. "Region-based Convolutional Neural Network" is abbreviated as R-CNN. There are two steps to the core concept. It starts by identifying a manageable number of bounding-box object region candidates via selective search. After that, each region's CNN features are extracted separately for classification. This is perfect for a CAPTCHA cracking model since most photos contain an item(s) that must be detected in order to identify whether or not the image contains that object. On its own however, the R-CNN model is expensive and slow. This is why “faster” versions of this model were developed, to provide a viable real-time solution to an interesting problem.

Object Detection

Object detection is a critical computer vision problem that detects occurrences of certain visual objects in digital pictures. The purpose of object detection is to create computational models that answer the most basic question that computer vision applications have: "What things are there and where are they?" The fast advancements of deep learning algorithms have substantially boosted the pace of object detection in recent years. With deep learning networks and the computing power of GPU’s, the performance of object detectors and trackers has greatly improved, achieving significant breakthroughs in object detection.

One-stage vs two-stage deep learning object detectors

In general, deep learning-based object detectors extract features from the input image or video frame. An object detector solves two subsequent tasks: Find and arbitrary number of objects (possibly zero); and classify every single object and estimate its size with a bounding box. To simplify the process, you can separate those tasks into two stages. Other methods combine both tasks into one step (single-stage detectors) to achieve higher performance at the cost of accuracy.

Two-stage detectors: In two-stage detectors, deep features are utilized to suggest approximate object areas before they are used for classification and bounding box regression for the object candidate. The maximum detection accuracy is achieved by two-stage algorithms, although they are often slower. The performance is not as strong as one-stage detectors due to the multiple inference stages per image. The currently implemented solution of Faster R-CNN + NASNet is an example of a two-stage method.

One-stage detectors: One-stage detectors predict bounding boxes over the images without the region proposal step. Because this procedure takes less time, it may be employed in real-time applications. The disadvantage is that they have difficulty detecting irregularly shaped objects or groups of small objects, which should not be a problem with CAPTCHA images. YOLO, SSD, and RetinaNet are some of the most popular one-stage detectors.

Artificial Neural Network

A short neural network has also been tested in ImageCAPTCHA branch under Rayvinder file