6G6Z0048 Artificial Intelligence: 1CWK100

Section 1: Longlisting

In this section I shall highlight relevant AI algorithms and concepts that relate to image classification problems, starting with three different Machine Learning approaches.

Supervised Learning is a popular AI methodology that consists of four stages -data collection, data preparation, training the model, and evaluation. SL uses algorithms that are trained on labelled datasets, of which we know from Founder 2, the company has. The goal being to learn patterns and be able to classify new data into categories, created from the algorithm's predictions. In this case, auction image categorisation.

Unsupervised Learning is like the prior, but with a focus on unlabelled data. Again, Founder 2 stated that they have a large amount. Using this data can have benefits such as identifying patterns for better categorisation unknown to labelled data. This approach can lead to incorrect results and require more manual clean up.

Semi-Supervised Learning combines the best of both. Utilising a smaller set of labelled and a larger amount of unlabelled historical auction images data, the algorithm of choice can learn even more ways to achieve accurate predictions. Usually this is used when we don't have access to a large portion of labelled data.

Feature space is the area where data points from out dataset are represented. A data point might be an image. In feature space, algorithms learn patterns from data and can make predictions on new data from this.

Feature Extraction is the transformation of raw data to relevant features the algorithms can understand. We can extract colours from images or edges of items.

Classification is a supervised learning concept of whereby a model predicts categories for new data, being new uploaded images in this case. Classification will categorise items in images based off the labels we define in our dataset.

Regression is typically used for numerical and/ or binary output. The company may want to use this technique in a later implementation to distinguish the condition of the items uploaded –damaged or undamaged.

Clustering is an unsupervised learning task where the chosen algorithm groups similar data points together in feature space. It uses unlabelled data to categorise based on structures and patterns. This can be effective with image categorisation -just what we need.

Hyperparameters are variables we set prior to training a model. The hyperparameters for one model can be completely different for another. In neural networks, a hyperparameter is the activation function. Activation functions are used to ensure non-linearity -the ability to learn complex relationships resulting in better predictions. The batch size is a hyperparameter we'll use for image classification, the number of training examples. Tuning hyperparameters is key to model success and requires trial and error. Changing them and tuning one way may lead to success but another way may do the opposite.

Overfitting can occur when a model learns exact training data. When a model learns data "too well", it is bound to that data, understanding those characteristics specifically, resulting in the lack of recognition of new data that doesn't match 1:1. If we feed our model with 500 images of which 400 of them are a car of the same model and colour, it won't be able to recognise different cars as all it knows is the car we provided. We can tune our hyperparameters to ensure overfitting doesn't occur.

Underfitting is the opposite. When a model isn't complex enough to adequately learn data and make appropriate predictions, it is unfit for its purpose. If we use a model that can't appropriately process image data, it will not be able to make further predications on new uploads. Underfitting can also occur with insufficient data. From what

Founder 2 mentioned, we have a large dataset of images, so underfitting is unlikely, although we still have a risk of the data being low resolution and not enough data within the image. Again, we tune hyperparameters to prevent underfitting.

K-Nearest Neighbours (k-NN) is a ML algorithm that makes use of the closest data points in feature space -the majority for classification and the average for regression. Finding the most common neighbours results in a label prediction for each new item image.

Artificial Neural Networks (ANNs) would be a good model to select as it operates most similar to the human brain - recognising images. By splitting up an image of an auction item in pixels, the model makes weightings of confidence for each, reaching a final prediction. ANNs splits data into neurons. These neurons are structured in an input layer. The neurons connect to many hidden layers of neurons. Here deep learning occurs. Neurons are activated as the model trains -this is handled by activation functions. Eventually the maths that occurs here culminate to an output on the output layer, being a prediction of an item in our case.

Activation functions decide whether a neuron should be activated or not. These functions calculate the weighting of a node. There are many types of activation functions: linear, non-linear, sigmoid, ReLU.

The cost function occurs after a prediction is made by the model designed to calculate the difference in the predicted values and the target values – "assesses the disparity between the predicted probability distribution over different classes for a given image, generated by the neural network, and the ground truth labels associated with that image".

Back propagation is a neural networks concept for adjusting parameters -weights and biases. The model will train, making and following connections between nodes, ending in a result -this is the forward pass. This result will have a weighting on how accurate it is. The model will send a signal back through the network to figure out the difference between its prediction and the target value -the compute loss. If the prediction is too high the weights of starting nodes and connections will decrement -weight update. The biases and weight pass through the activation function of a neuron which then decides to activate or not.

Gradient descent is an optimisation algorithm used to minimise the cost or loss function. This is used particularly with image classification and neural networks to adjust weights and improve accuracy predicted labels and actual class labels.

Convolutional Neural Networks (CNNs) are specialised version of ANN, designed with layers that are purposely engineered for image classification. Designed for row-like structure, CNNs are good at recognising edges and texture, for example, two things that may be vastly different for each auction item. Applying convolution, the model can learn segments of an image to then piece together for the greater image. This feature extraction and architecture significantly improves performance over traditional ANNs.

Support Vector Machines (SVM) are another supervised learning algorithm that uses a 'hyperplane' to separate data points in feature space that are of different classes. SVM find support vectors and determine the boundary (the points closest to the plane). Images must first be converted into vectors where these vectors are used to find the hyperplane.

K-means is an unsupervised learning clustering algorithm which groups together similar data points in feature space. The clusters might share likeness in colour within image classification. Each cluster as a central point -a centroid, where new data will link to the nearest one.

Section 2: Analysis

Supervised Learning

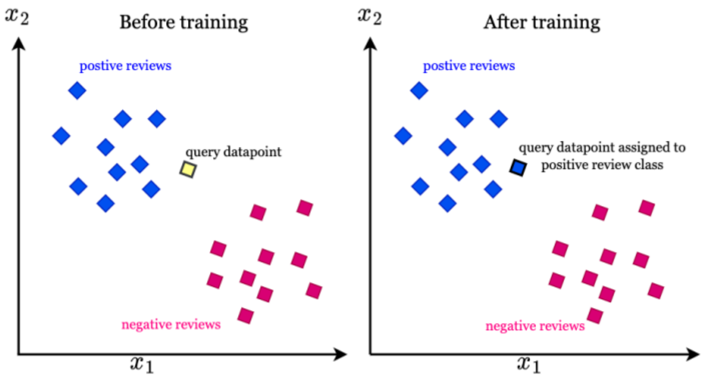
Discover a relationship between one special value and other values (Lecture 2). Founder 2 stated that the company has a “large collection of historical auction images with classification labels”. Supervised Learning is a ML methodology that trains a model from labelled data, data that can reach high prediction accuracy. This can have ethical issues as most labelled data is currently from larger markets. Founder 4 pointed out their smaller markets have less data and therefore their items may be incorrectly classified.

Stage 1 is already complete as we have labelled dataset, greatly reducing preparation time.
Data preparation is stage 2 where we sort the data, spot mistakes, and fix. Founder 2’s moderation team could do this, however this does require manual checking and time.
The third stage is training the model. Once we have chosen our model, we must feed it with the data we collected and prepared -the images. The dataset needs to be split into two categories -testing data and training data. Training the model with 70% of the labelled image dataset (target feature values) and then testing with the remaining 30% will allow the model to learn with a large set of data whilst leaving enough to assess predication accuracy and performance. During this fitting phase, we need to be wary of overfitting. If we train the model with many similar items, it won’t be able to differentiate effectively and be bound to that data.
Finally, evaluation stage. Once trained, use testing data to discover the accuracy of predictions. If as humans we recognise a car, but the model doesn’t, we must tune the hyperparameters to hopefully identify correctly next iteration. While this trial and error can take time, it leads to better results. We must also take note of other performance such as training and prediction times. The company desires the feature be implemented soon and if training takes too long, they will continue to lose business. Likewise, if the model is slow to classify, the user experience hasn’t been improved. We must evaluate whether overfitting or underfitting has occurred. If we use a supervised learning algorithm that continues to underfit, we may be best using the larger set of unlabelled data with a different methodology.

KNN

The K-Nearest Neighbours model has already been experimented with by Founder 1. She has a desire for a simple and efficient solution. While k-NN can offer the former, it struggles with efficiency of large datasets. Considering the data consists of thousands of images, this may not be the best model, but it can work. By grouping data points together in feature space based on their nearest neighbour, the model calculates distances between a new data point and the K closest trained data points. In classification, it predicts the majority class but in regression, it predicts the average value. A new image of headphones is likely to be in the neighbourhood of other headphones and link together. Each image can be represented as a point in feature space and the dimensions we can set to pixel values. When finding the neighbours, we must consider how many we want to use -the K value, the model’s main hyperparameter. Tuning K is found through trial and error. We could set K to half the average number of items in each category. That could be too large and create complexity and lead to overfitting, or it could be too small, and the model will fail to find similar items. Too large of a K can also lead to bias, a key issue Founder 4 is keen to avoid. Therefore, careful selection of K is essential. Implementing cross-validation helps finding the choice of K, increases performance, and shortens time.

A k-NN model was created and tested (My Learnings in AI ML, no date) for 3000 images, and proves a 52% accuracy. While not a high score, it completed the task using only 9MB of memory, showing extreme efficiency and the ability to run on low powered devices.

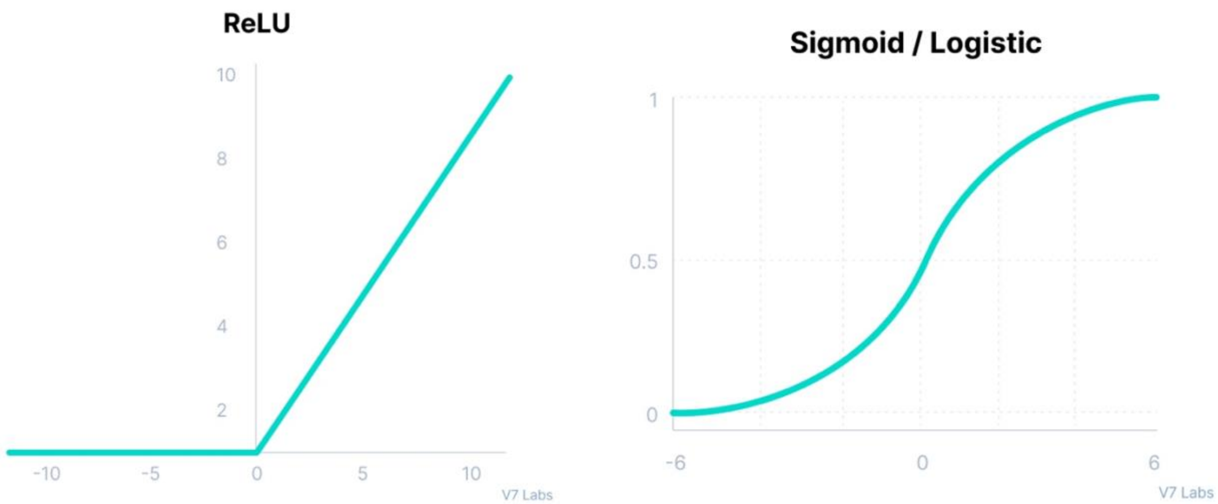


Example KNN. Not related to linked test.
https://intuitivetutorial.com/wp-content/uploads/2023/04/knn-1.png

ANN CNN

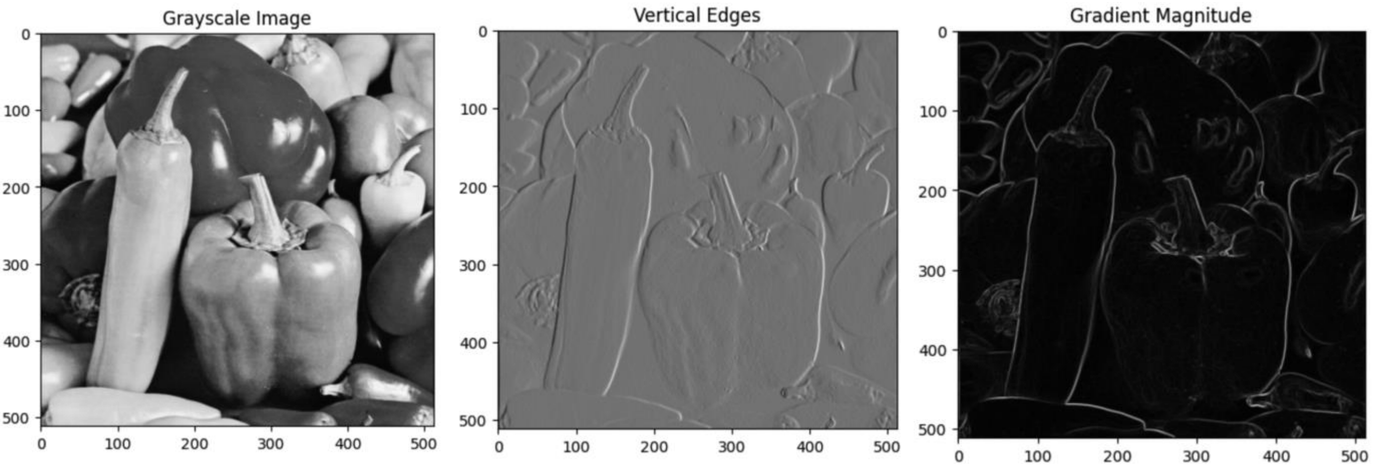
As we are attempting to predict items from images, an ANN will separate an image into pixels. These pixels are classed as nodes/neurons. These nodes are arranged into the three layers. The separated pixel values may not be enough to recognise patterns, therefore the hidden layers will apply feature extraction to make a result. I like to visualise layers as walls with doors. Each door can be opened and lead to the next. Doors will be unlocked by an activation function. Imagine walking through the model, entering each unlocked door, wall by wall, following a path called the forward pass, reaching the final door and making a prediction of what we believe the journey led us to. The prediction may be right, wrong, or close. Regardless, we must send a signal back informing the performance. This is back propagation. It will change the weightings of the paths we took to get to the end. A performance factor is the cost function -the difference between the predicted values and the true values. With new weightings, the model will be able to minimise the cost and make more accurate predictions each time, through gradient descent. Gradient decent can become slow and cumbersome to update weightings -the vanishing gradient problem. To speed this up, we apply an activation function.

There are multiple activation functions such as sigmoid and ReLU. The activation for every neuron in each hidden layer is calculated as the weighted sum of the activations from previous layer, plus the bias (3Blue1Brown, 2017). We pass the result through an activation function. ReLU is most popular due its simplicity, a factor Founder 1 wishes for. ReLU is fast and applies non-linearity, allowing a neural network to learn complex patterns and relationships which image data is renowned for. Sigmoid attempts to avoid the vanishing gradient problem found in backpropagation but ReLU is far more efficient. Sigmoid is best with binary problems but ReLU is best for classification.



https://www.v7labs.com/blog/neural-networks-activation-functions

There are many ways to train neural networks. One approach being to decolourise the images in pre-processing. All images have RGB colour values and storing these increases complexity. Converting an image to greyscale reduces the complexity. This binary transformation would work well for a sigmoid function but again, it's not as ideal for image classification. When greyscale, the hidden layers can feature extract edges of items for prediction later. Shown below from the Lab 6B code.



ANN is the best performer in terms of accuracy. This ANN trained by Kaggle (NITESH YADAV, no date) achieves an 88.8% accuracy.

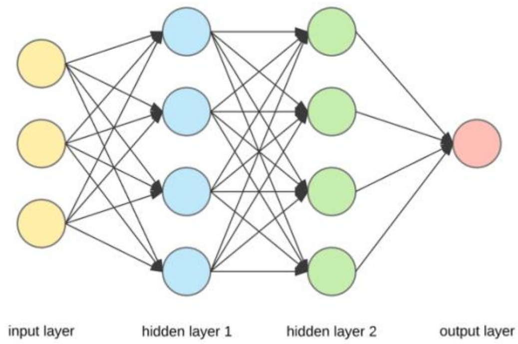
An alternative ANN (Periyasamy N, Thamilselvan P and J. G. R. Sathiaseelan, March 2015) attains a 98% accuracy while predicting in 4-5 seconds. Incredible score in a reasonable time.

Training a model using images is more computationally expensive than other data due to file sizes and complexity. Users may upload a 4K images -8.3 million pixels images. Training the models on many pixels for each image would take a long time in a traditional ANN. CNN is a solution designed for structured grid data, like images. A method to improve efficiency is to reduce the image size. If we compress every image to a more manageable resolution that still retains eligibility, a CNN can process at far greater speeds. We can make a CNN cluster pixels into groups. Instead of processing individually, we can process a number together, further reducing processing -this process is called pooling. When compressing, we must balance detail for training, with a small enough size to process in a reasonable time.

CNNs are complex, but Founder 1 has a desire for a simplicity. She also wants to prioritise free and open-source projects. TensorFlow and PyTorch are both widely used, free frameworks that we can use to work with our dataset that speed up implementation. We can also use these frameworks with pretrained models. A carefully chosen model trained on data like ours is needed, but time and ability to train our own is mitigated. If we find a model that uses an ImageNet7 dataset of images like ours, we could use that. Both options make implementation simpler as the heavy work is complete, we just need to ensure interoperability with our dataset.

When using TensorFlow for blood cell images for a CNN (Analytics Vidhya, June 2023) this model achieves an 86% accuracy. A slightly lower score than a typical ANN, but at far greater efficiency.

Another using 4000 images of cats and dogs (S. Panigrahi, A. Nanda and T. Swarnkar, 2018) achieves a classification accuracy of 88.3%.

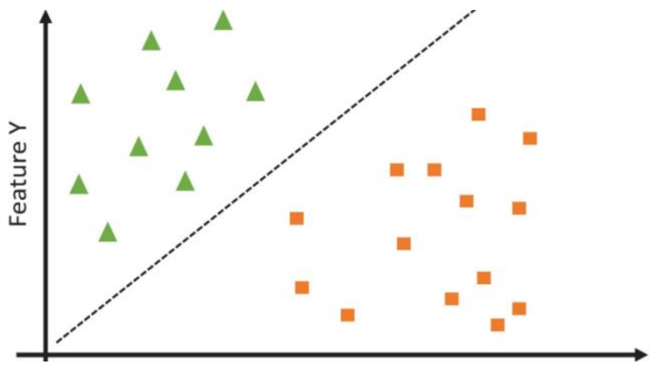


Example neural network.
*https://miro.medium.com/v2/resize:fit:1358/1*Gh5PS4R_A5drl5ebd_gNrg@2x.png*

SVM

Images and Support Vector Machines can work well, but a few steps are required, starting with feature extraction. Images use colour so I'll chose flattening. By converting the images into 1D vectors, the model can make use of them. Each image is a matrix of pixels -we can concatenate the matrix into a vector. The company has plentiful labelled data which is a benefit as we could manually label all unlabelled data, but we don't need to. Before training our SVM, the vectors need to be normalised in the form such that the values range from 0 to 1. This min-max scaling ensures all features contribute equally to the model and prevents dominant features from affecting the output -a benefit for model accuracy but a negative for implementation complexity. As a SVM is trained, it will automatically learn the hyperplane that best works for separation between classes. All new test data will also need to be converted, but now the model will classify each image based on its proximity to the plane.

SVMs are not the best approach for image classification. They are computationally expensive and are bad with large datasets, of which we have. More data is more noise, it may class non-similar items together; it's a challenge to separate the noise. Founder 3 stated a desire to move compute onto user devices, of which most will be on mobile phones, devices that can't handle heavy tasks SVMs provide. Whilst they can be accurate enough, this SVM trained (GeeksForGeeks, 2023) provides an accuracy of 59% showing a large proportion is inaccurate.



Example SVM. Not related to linked test from GeeksForGeeks.
https://images.spiceworks.com/wp-content/uploads/2022/09/02134804/Diagram-depicting-SVM-example-with-hyperplane-for-classification-problem.jpg

Unsupervised Learning

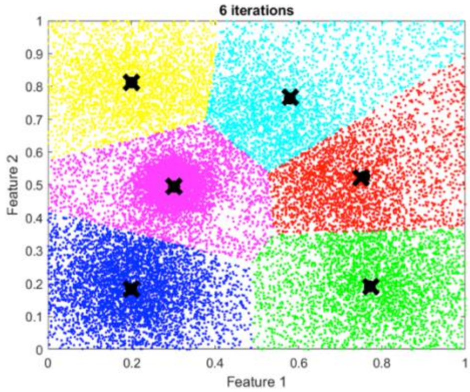
Seek out useful hidden structure by examining relationships between all values in a dataset (Lecture 14). Unsupervised Learning makes use of unlabelled data -which Founder 2 identified an even larger repository of. The goal is for a model to find patterns and relationships on its own, without user guidance -this is why we use unlabelled data. It'll find hidden patterns which humans and labelled models are unable to spot e.g. colours, and textures. We can apply clustering algorithms that recognise these patterns to increase performance. Labelled data may be incorrect and only consist of a few taxonomies, that issue is irradicated here.

While using unlabelled data cuts out manual categorisation time, it does lead to problems. The algorithms can struggle to differentiate patterns in noisy data. Without a label, it is prone to confuse structures with noise. An example is a watch on a person's wrist. The model might focus on data in the background (noise) instead of understanding the watch itself. With labelled data, we can clearly define the watch. This also leads to subjectivity in interpretation. Conversely, using unlabelled data eliminates the need for human moderators to label items, allowing models to interpret lesser-known items from smaller markets where there aren't pre-categorised images.

K-means

A clustering algorithm within unsupervised learning, K-means. We define the number of clusters -we chose K. The algorithm can cluster colours of pixels in an image. For this example, we may want to identify small detail, giving K a small value. We may want to cover large areas and more general structures, as seen in the examples below. Choosing K can be solved by using methods like cross-validation or elbow plot. Incorporating these methods adds complexity to implementation but are easy enough to apply and leads to enhanced results. You'll notice a centre point; this is the centroid. Each cluster has a centroid -the average position of all points. New data can be compared to the centroid rather than all data points. Whichever centroid is closest is where the data will be classed, greatly improving performance. This can however lead to a small set of categories which new data, that is unlike much other, may be classified incorrectly. Founder 4 is wary of this as they have limited access to smaller markets data, items that may be grouped where they don't belong, causing upset.

The K-means is effective at classifying images, as the model implemented by machinelearningmastery.com (Machine Learning Mastery, 2023) achieves an accuracy of 70.9% showing a strong ability to predict correctly.



Lecture 14. Example K-means. Not related to linked test above.

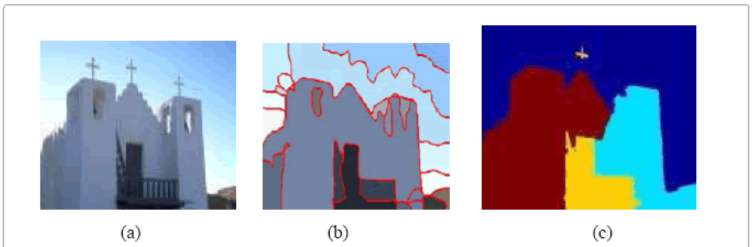


Figure 1: (a) is the original image; (b) and (c) are the segmentation results.

Example K-means. Not related to linked test above.
*https://miro.medium.com/max/699/1*XbIyaCstQ2-k0dSmdDYhog.png*

Semi-supervised Learning

This approach utilises both labelled and unlabelled data, but mostly the latter. Using both types, results in less data wasted. However, it can still waste labelled data if we have an abundance, as it largely focuses on unlabelled. If the unlabelled data doesn't hold much useful information, the models won't be effective. Implementing a model that incorporates both can on the other hand, can result in better predictions by leveraging knowledge from labels and no labels. We should first train a model with labelled data and secondly feed the training unlabelled data which can hopefully recognise items more easily and generally. Utilising both datasets reduces bias but implementing a semi-supervised algorithm can often be more complex than supervised learning approaches.

Section 3: Recommendation

I suggest the company implements a Convolutional Neural Network (CNN) model, as a classification supervised learning task. This is for a few reasons but largely for their effectiveness in image classification. Although Founder 1 had started a k-NN implementation and has a fondness for its simplicity and efficiency, the benefits of CNNs out way the ones for k-NN.

K-NN is possible to run on local devices, a preference of Founder 3, but is far less accurate with predictions. While a CNN would need to be performed in the cloud from an IaaS, increasing cost, the service has a history of AI related work and negotiations are possible through the good working relationship.

While CNNs can be complex, they are made considerably easier to implement and understand through pre-trained models and frameworks that speed up the process. These frameworks have good documentation and are easy to learn and scale, as Founder 1 wants. As the company has a large collection of labelled historical auction images, we can easily train a neural network using a supervised learning approach. Although they can use a pretrained model, I recommend they train with their own data, as the model will be trained with their exact images -images that are already prepared. This will increase time, but it allows Founder 1 to have full control and understand the solution wholly.

Although most pre-labelled data are from larger markets, Founder 2's moderation team could label additional items from smaller demographics. This way the model learns and contains more information, reducing Founder 4's concerns of imbalance, and the dataset's lack of variety. As this team exists, it's very likely most labelled data is correct and wouldn't require a lot of work to check. With this labelled data, they have a strong foundation to begin training the model. Utilising the vast quantity of unlabelled images would allow us to use the K-means algorithm and not waste as much data, however CNN provides better results while being more efficient, with labelled data.

Using an ANN typically provides more accurate predictions, but they come at a higher computational cost and therefore a financial cost for improved cloud computing. CNNs use the foundation built by an ANN while being more efficient at image classification tasks. ANNs achieve high accuracy but CNNs come very close.

In conclusion, I believe a CNN is the best approach for the company's image classification task.

Reference list

Lecture 2: Supervised Learning I
Lecture 14: Unsupervised Learning I

3Blue1Brown (2017). *Gradient descent, how neural networks learn | Chapter 2, Deep learning.*
https://youtu.be/IHZwWFHWa-w?si=4di_cVE9IjRotWkz

My Learnings in AI ML (no date) *First Image Classifier -KNN*
Available at: https://mylearningsinaiml.wordpress.com/deep-learning/dl4cv/first-image-classifier/
(Accessed: 29th November 2023).

NITESH YADAV (no date) *Image Classification with ANN*
Available at: https://www.kaggle.com/code/niteshyadav3103/image-classification-with-ann
(Accessed: 29th November 2023).

Periyasamy N, Thamilselvan P and J. G. R. Sathiaseelan, "A comparative study of ANN, k-means and AdaBoost algorithms for image classification," 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 2015, pp. 1-5, doi: 10.1109/ICIIECS.2015.7193067.
(Accessed: 4th November 2023).

Analytics Vidhya (June 2023) *Image Classification Using CNN*

Available at: https://www.analyticsvidhya.com/blog/2021/01/image-classification-using-convolutional-neural-networks-a-step-by-step-guide/
(Accessed: 30th November 2023)

S. Panigrahi, A. Nanda and T. Swarnkar, "Deep Learning Approach for Image Classification," *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)*, Changsha, China, 2018, pp. 511-516, doi: 10.1109/ICDSBA.2018.00101.
(Accessed: 4th December 2023).

Geeks for Geeks (March 2023) *Image classification using Support Vector Machine (SVM) in Python*
Available at: h https://www.geeksforgeeks.org/image-classification-using-support-vector-machine-svm-in-python/
(Accessed: 30th November 2023)

Machine Learning Mastery (2023) *k-means clustering for image classification using opencv*.
Available at: https://machinelearningmastery.com/k-means-clustering-for-image-classification-using-opencv/
(Accessed: 1st December 2023).

Appendix 1: LLM conversations

Below are all questions I asked ChatGPT.

Prompt	Response
does cost funcition assesses prediction accuracy?	Yes, the cost function assesses the difference between the predicted values and the true values (ground truth) and provides a measure of how well the model is performing. The goal during training is to minimize this cost function. In the context of supervised learning, the cost function helps quantify the error or discrepancy between what the model predicts and what is actually observed in the training data. The lower the cost, the better the model's predictions align with the true values. Minimizing the cost function is a crucial step in training a machine learning model, including neural networks. Different types of tasks (e.g., classification, regression) and model architectures may require different cost functions. For example: • In regression problems, where the goal is to predict continuous values, the Mean Squared Error (MSE) is commonly used as a cost function. • In binary classification problems, Binary Crossentropy is often used. • In multiclass classification problems, Categorical Crossentropy is commonly employed. The process of minimizing the cost function involves adjusting the model's parameters through optimization techniques like gradient descent. The model iteratively learns from the training data, updating its parameters to reduce the error and improve its predictive accuracy.
Can you explain what the cost function assesses in relation to image classification in one sentence	In image classification, the cost function assesses the disparity between the predicted probability distribution over different classes for a given image, generated by the neural network, and the ground truth labels associated with that image.
how to make an svm more accurate	Improving the accuracy of a Support Vector Machine (SVM) involves various strategies related to data preprocessing,

model tuning, and optimization. Here are some common approaches:

1. **Feature Scaling:**
    - Standardize or normalize your input features to ensure that they are on a similar scale. SVMs are sensitive to the scale of input features, and standardization can help improve their performance.
2. **Hyperparameter Tuning:**
    - Experiment with different values for hyperparameters like the regularization parameter (C) and the choice of kernel (linear, polynomial, or radial basis function - RBF).
    - Use techniques like grid search or randomized search to systematically search through the hyperparameter space.
3. **Kernel Selection:**
    - Choose an appropriate kernel for your problem. In many cases, the RBF kernel is a good default choice, but depending on the nature of your data, linear or polynomial kernels might be more suitable.
4. **Cross-Validation:**
    - Utilize cross-validation techniques (e.g., k-fold cross-validation) to assess the model's performance and generalize well to unseen data. Cross-validation helps in obtaining a more robust estimate of the model's accuracy.
5. **Class Imbalance Handling:**
    - If your dataset has class imbalance, consider strategies like oversampling the minority class, undersampling the majority class, or using techniques like SMOTE (Synthetic Minority Over-sampling Technique).
6. **Outlier Detection and Removal:**
    - Identify and handle outliers in your dataset, as outliers can impact the decision boundary of the SVM. Consider removing or adjusting outliers to improve the model's generalization.
7. **Feature Engineering:**
    - Carefully select and engineer features that are most relevant to your problem. Feature engineering can improve the discriminatory power of the SVM.
8. **Data Cleaning:**
    - Ensure that your dataset is clean and free from errors. Remove or correct any inconsistencies, missing values, or anomalies in your data.
9. **Ensemble Methods:**
    - Explore ensemble methods such as bagging or boosting with SVMs. Techniques like Random Forests with SVMs as base learners can sometimes improve overall performance.

	10. **Advanced SVM Variants:** • Consider trying advanced SVM variants such as Nu-SVM or SVM with a different kernel trick based on the nature of your data. Remember to carefully validate the performance improvements using proper evaluation metrics, and be cautious about overfitting to the training data. The effectiveness of these strategies may depend on the specific characteristics of your dataset and the nature of the problem you're solving.
i have a set of labelled images, theyre the target feature values arent they	Yes, in the context of supervised learning, a set of labeled images typically consists of both the input features (the images) and the corresponding target feature values (the labels). Each image is associated with a specific label or class that indicates the category or class to which the image belongs.
Can you briefly explain hyperparameters so I know my knowledge is crroect	Certainly! In machine learning, hyperparameters are configuration settings for a model that are not learned from the data but are set before the training process begins. These parameters influence the learning process and the structure of the model but are not updated during training. The values of hyperparameters are typically set by the machine learning engineer or data scientist based on their domain knowledge, experience, or through systematic tuning. Here are some common examples of hyperparameters in machine learning models: 1. **Learning Rate:** • In optimization algorithms like gradient descent, the learning rate determines the size of the steps taken during the parameter updates. It is a crucial hyperparameter that affects the convergence of the model. 2. **Number of Hidden Layers and Neurons:** • For neural networks, the architecture is defined by hyperparameters such as the number of hidden layers, the number of neurons in each layer, and the activation functions used. 3. **Regularization Strength:** • Regularization hyperparameters control the amount of regularization applied to the model. This includes L1 and L2 regularization parameters. 4. **Kernel Size and Stride:** • In convolutional neural networks (CNNs), hyperparameters like kernel size and stride determine the size of filters used for convolution operations. 5. **Number of Trees and Tree Depth:** • In ensemble methods like random forests or gradient boosting, hyperparameters include the number of trees in the ensemble and the maximum depth of each tree. 6. **Batch Size:**

	• For training deep learning models, the batch size represents the number of training examples used in one iteration. It is a hyperparameter that affects the efficiency of training. 7. **Number of Clusters:** • In clustering algorithms like k-means, the number of clusters is a hyperparameter that must be specified before training. 8. **C in Support Vector Machines (SVM):** • In SVM, the regularization parameter �C determines the trade-off between achieving a low training error and a low testing error. Hyperparameter tuning involves experimenting with different values for these settings to find the combination that results in the best performance on a validation set or through cross-validation. Proper tuning of hyperparameters is essential to achieve the best possible model performance.
overfitting is likely to occur when we feed the model too much similar data isnt it?	Yes, overfitting is more likely to occur when a model is trained on too much similar or redundant data. Overfitting happens when a machine learning model learns the training data too well, capturing not only the underlying patterns but also the noise and specificities of the training set. As a result, the model may perform well on the training data but generalize poorly to new, unseen data. When a model is trained on an abundance of similar data, it can memorize the specific details and variations in the training set rather than learning the true underlying patterns.
give me 4 bullet points, one as an advantage for ANN and a disadvantage for ANN and the same for CNN	**Artificial Neural Networks (ANNs):** **Advantage:** - **Versatility:** ANNs are highly versatile and can be applied to a wide range of tasks, including regression, classification, and pattern recognition. They can learn complex relationships in data and adapt to different types of problems. **Disadvantage:** - **Computational Intensity:** Training large ANNs can be computationally intensive, especially deep networks with a large number of parameters. This can require powerful hardware and substantial computing resources. **Convolutional Neural Networks (CNNs):** **Advantage:** - **Spatial Hierarchies for Image Data:** CNNs excel at capturing spatial hierarchies and local patterns in image data. Through convolutional layers and pooling operations, CNNs can efficiently learn hierarchical features, making them well-suited for computer vision tasks. **Disadvantage:** - **Data Efficiency for Non-Image Data:** CNNs are specifically designed for grid-like data, such as images. For non-grid data, the architecture might not be as efficient, and alternative architectures like recurrent neural networks (RNNs) or transformers might be more appropriate.

	These points highlight some of the strengths and weaknesses of ANNs and CNNs, but it's important to note that the choice between them depends on the specific characteristics of the data and the nature of the task at hand.
what are some of the methods i can use to chose k in k-means? I don't need explanations, just reply in one sentence	Common methods for choosing the number of clusters (K) in K-means include the elbow method, silhouette analysis, the Davies-Bouldin index, and domain knowledge.
reply with yes or no. Back propagation is where we go through the model, reach the output and send a signal back informing performance right?	Yes.
reply with yes or no. CNN is a image specialised ANN model. Its not a whole new model and its not within the layers of an ANN, but it is a form of an ANN	Yes.
brief difference between classification and regression	**Classification:** • **Definition:** Classification is a type of supervised learning task where the goal is to assign predefined categories or labels to input data based on its characteristics. • **Output:** The output is a discrete category or label. • **Examples:** Examples include email spam detection (classifying emails as spam or not spam) or image classification (assigning objects in images to predefined classes). **Regression:** • **Definition:** Regression is another type of supervised learning task, but it involves predicting a continuous numerical value rather than a category or label. • **Output:** The output is a continuous value. • **Examples:** Predicting house prices based on features like square footage, predicting temperature, or predicting sales revenue. In summary, classification and regression are both supervised learning tasks, but they differ in the nature of the output they produce.
can you do classification in unsupervised learning?	Traditionally, classification is considered a supervised learning task where the algorithm is trained on labeled data with known outcomes or categories. However, there are certain techniques within unsupervised learning that can be used to achieve tasks that are conceptually similar to classification. These techniques include clustering, manifold learning and anomaly detection. It's important to note that while these techniques can be used to discover patterns or structures in the data that resemble classes, they do not strictly adhere to the conventional definition of classification. The absence of explicit, predefined class labels distinguishes these approaches from traditional supervised classification tasks.

Appendix 2: Source code

LAB 6B - ANN Decolourisation.

```
import matplotlib.pyplot as plt
from skimage import io, color, feature, filters, exposure
from scipy import ndimage as ndi
from skimage.transform import resize
```

```
import numpy as np

# Load the image
image = io.imread('peppers.png')
print("Original image shape:", image.shape)

# Show the colour image
plt.imshow(image)
plt.title('Original Image')
plt.show()

# Convert to grayscale
gray_image = color.rgb2gray(image)
print("Grayscale image shape:", gray_image.shape)

# Show the grayscale image
plt.imshow(gray_image, cmap='gray')
plt.title('Grayscale Image')
plt.show()

# Convolution with a vertical edge detection filter
vertical_filter = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
edges_vertical = ndi.convolve(gray_image, vertical_filter)

# Show the vertical edges
plt.imshow(edges_vertical, cmap='gray')
plt.title('Vertical Edges')
plt.show()

# Convolution with a horizontal edge detection filter
horizontal_filter = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])
edges_horizontal = ndi.convolve(gray_image, horizontal_filter)

# Show the horizontal edges
plt.imshow(edges_horizontal, cmap='gray')
plt.title('Horizontal Edges')
plt.show()

# Calculate the gradient magnitude
edges_magnitude = np.sqrt(edges_vertical**2 + edges_horizontal**2)

# Normalize the gradient magnitude to the range [0, 1] for visualization
edges_magnitude_normalized = edges_magnitude / np.max(edges_magnitude)

# Show the gradient magnitude
plt.imshow(edges_magnitude_normalized, cmap='gray')
plt.title('Gradient Magnitude')
plt.show()

# HOG feature extraction and visualization
```

```
fd, hog_image = feature.hog(gray_image, orientations=8, pixels_per_cell=(16, 16),
visualize=True, block_norm='L2')

# Show HOG image
plt.imshow(hog_image, cmap='gray')
plt.title('HOG Features Visualization')
plt.show()
```