Neural Network Models for Object Recognition

Credits

The template has been provided by slidesgo
Machine learning icons created by Ranah Pixel Studio - Flaticon
Vision icons created by Freepik - Flaticon
Nlp icons created by Freepik - Flaticon

ARTIFICIAL INTELLIGENCE & OBJECT RECOGNITION

- **Artificial Intelligence (AI)** is "...the science and engineering of making intelligent machines..." (IBM, N.D.)
- The AI market: $86.9 billion revenue in 2022. Estimated $407 billion revenue in 2027. (Haan, 2023)
- Impact of most technologies on jobs expected to be a net positive over the next five years (WEF, 2023)

AI MAIN FIELDS

Machine Learning

Giving the ability to machines to 'learn'

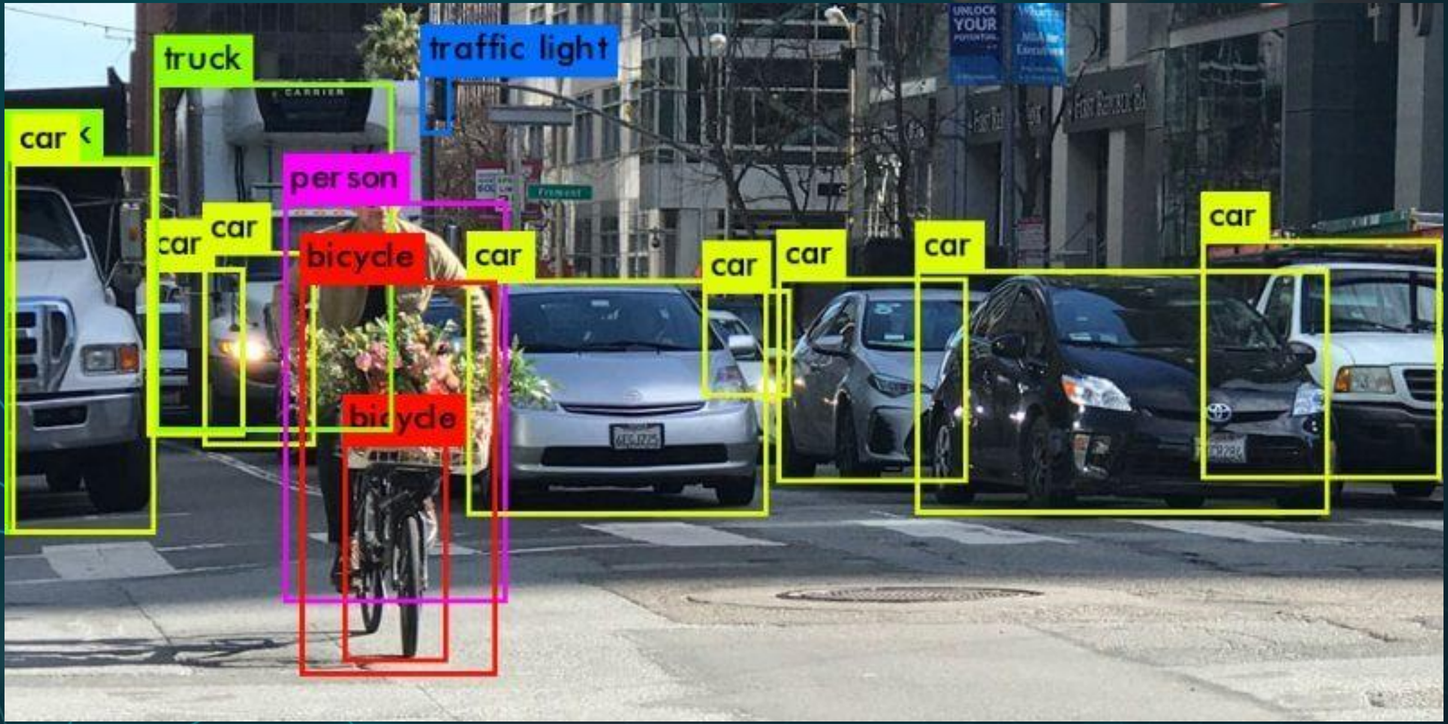
Computer Vision

Giving the ability to machines to 'see'

Natural Language Processing

Giving the ability to machines to 'understand'

OBJECT RECOGNITION (EXAMPLE)



Object detection and recognition, use case example (Azati, 2022)

THE TASK

- Our task
  - Train a neural network for object recognition with the CIFAR-10 image dataset

- The dataset
  - CIFAR-10 small images classification dataset by Keras
  - 60,000 images
  - 32*32 dimension RGB colour images making the shape of each image 32*32*3
  - 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
  - Pre-split by Keras into 50,000 training images & 10,000 test images

ARTIFICIAL NEURAL NETWORK (ANN) DESIGN

- Train / validation / test split

```
# load cifar10 in predefined train / test split
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# split the training data into training and validation sets
# set `random_state` to 0 to ensure the same split every time the code is ran
# 20% ie 10000 entries will be split from the training set into validation
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=0)
```

```
print(len(X_train))
print(len(X_test))
print(len(X_val))
```

```
40000
10000
10000
```

- Categorical Cross-Entropy loss function

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i$$

Categorical Cross-Entropy loss function (androidkt, 2023)

```
# early stop when model stops performing: https://keras.io/api/callbacks/early_stopping/
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
```

- Data pre-processing

```
# normalise pixel values to be between 0 and 1 by dividing by the maximum RGB value (255)
# this aids the neural network in processing the input images
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255
X_val = X_val.astype('float32') / 255
```

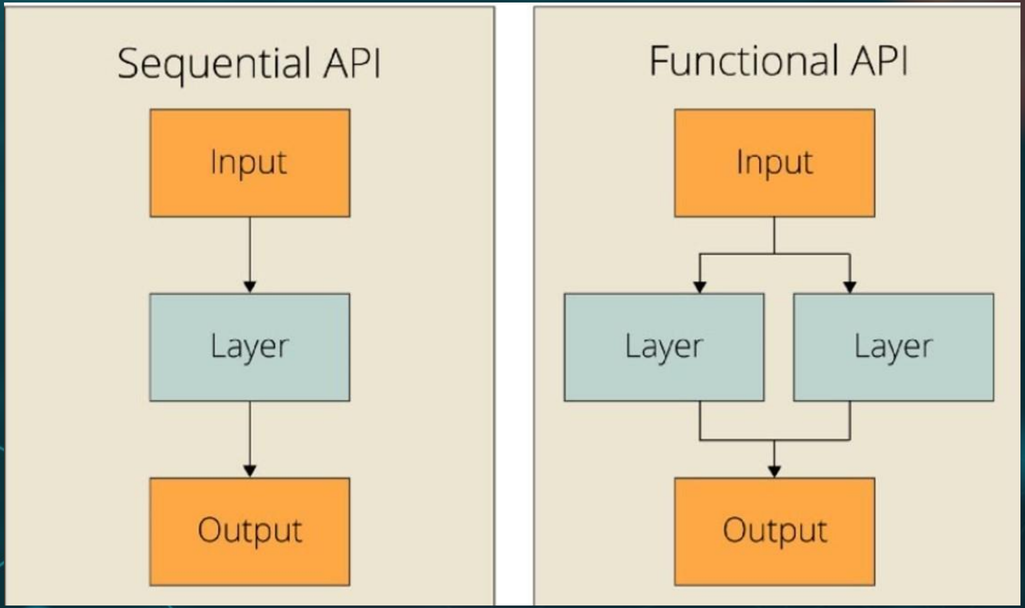
```
# the output of the neural network will be a probability distribution over the classes
# the labels are transformed to one-hot encoding to match this format
# allows for a more direct comparison between the neural network's output and the true labels
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)
```

ANN DESIGN

```
# instantiate the model
model = Sequential()
# input layer is created by Flatten
# set size and structure of inputs
model.add(Flatten(input_shape=(32, 32, 3)))
# set hidden layers
model.add(Dense(1000, activation='elu', kernel_initializer='he_uniform', bias_initializer='truncated_normal'))
model.add(Dropout(0.5))
model.add(Dense(1000, activation='elu', kernel_initializer='he_uniform', bias_initializer='truncated_normal'))
model.add(Dropout(0.5))
# output layer with 10 neurons and softmax activation function
# rule of thumb, number of neurons in output label is equal to number of classes/labels that you are predicting
model.add(Dense(10, activation='softmax'))
```

ANN METHODOLOGY (The model)

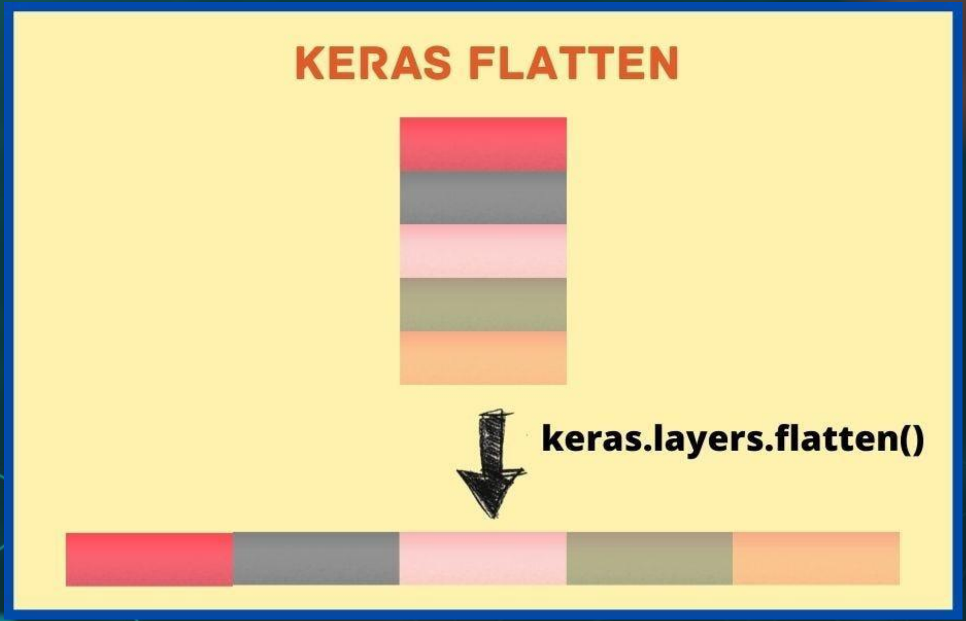
```
# instantiate the model
model = Sequential()
```



Sequential and Functional APIs architecture (Analytics Vidhya, 2022)

ANN METHODOLOGY (Flattening)

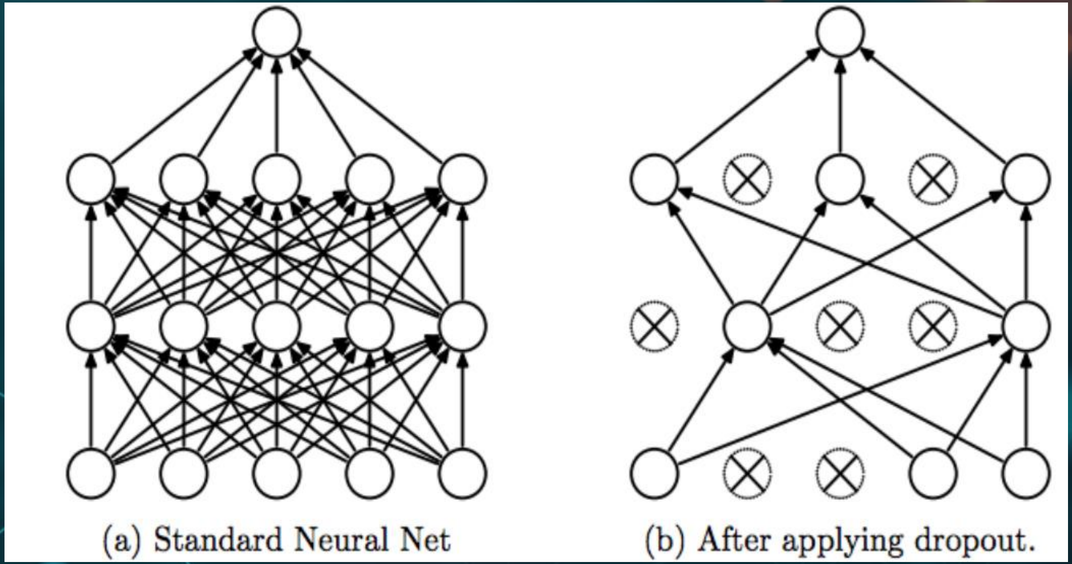
```
# input layer is created by Flatten
# set size and structure of inputs
model.add(Flatten(input_shape=(32, 32, 3)))
```



Keras flattening (McLean, 2021)

ANN METHODOLOGY (Hidden layers)

```
# set hidden layers
model.add(Dense(1000, activation='elu', kernel_initializer='he_uniform', bias_initializer='truncated_normal'))
model.add(Dropout(0.5))
model.add(Dense(1000, activation='elu', kernel_initializer='he_uniform', bias_initializer='truncated_normal'))
model.add(Dropout(0.5))
```



Dense layer with and without dropout layer (Srivastava et al., 2014)

ANN METHODOLOGY (Hidden layers cont.)

```
model.add(layers.Dense(  # Only Dense layer is used to connect every input with every neuron
 units=unit,
 kernel_initializer=weight,
 bias_initializer=weight2,
 kernel_regularizer=weightreg,
 activity_regularizer=weightreg2,
 activation=activ))
```

Iteration code snippet

Optimizer	Neurons	Activator	Kernel Initializer	Bias Initializer	Kernel Regularizaer	Activity Regularizer	Test Accuracy	Test loss
Adamax	50	selu	zeros	truncated_normal	None	None	0.502300024	1.43974018
Adamax	50	elu	zeros	he_uniform	None	None	0.501999974	1.43976223
sgd	50	elu	zeros	he_normal	None	None	0.50150001	1.43572009
Adamax	50	softplus	zeros	zeros	None	None	0.500800014	1.46310842
Adamax	50	elu	None	he_uniform	None	None	0.499900013	1.4597578
Adamax	50	elu	random_normal	ones	None	None	0.498400003	1.47205293
Adamax	50	elu	zeros	variance_scaling	None	None	0.498199999	1.45974982
Adamax	50	softplus	None	he_uniform	None	None	0.497999996	1.47181857

Iteration results snippet

ANN METHODOLOGY (Output, Compilation, Fitting, Results)

```
model.add(Dense(10, activation='softmax'))
```

Output layer

```
# initialise adam optimiser
model.compile(loss='categorical_crossentropy',
              optimizer="Adamax",
              metrics=['acc'])  # can set to accuracy, precision, or recall
```

Compilation

```
# train model
# update model's weights each time to minimise the loss function
# higher epochs can result in higher accuracy but more likely  to overfit
# weights are not adjusted by performance results from validation set, it's purely a benchmark
model.fit(X_train, y_train, batch_size=128, epochs=200, callbacks=[early_stopping], validation_data=(X_val, y_val))
```

Fitting

```
Epoch 108/200
313/313 [==============================] - 33s 106ms/step - loss: 1.0093 - acc: 0.6410 - val_loss: 1.2236 - val_acc: 0.5808
```

Final epoch for the selected model and related metrics

CONVOLUTIONAL NEURAL NETWORK (CNN)

- We realised that we were unlikely to get our ANN model to a satisfactory accuracy
- Convolutional Neural Networks (CNNs) are excellent at image classification (Sultana et al, 2018)
- CNNs are good for image classification because the convolutional layers extracts the features (Wang et al, 2020)
- CNN development was started in parallel to completing ANN development

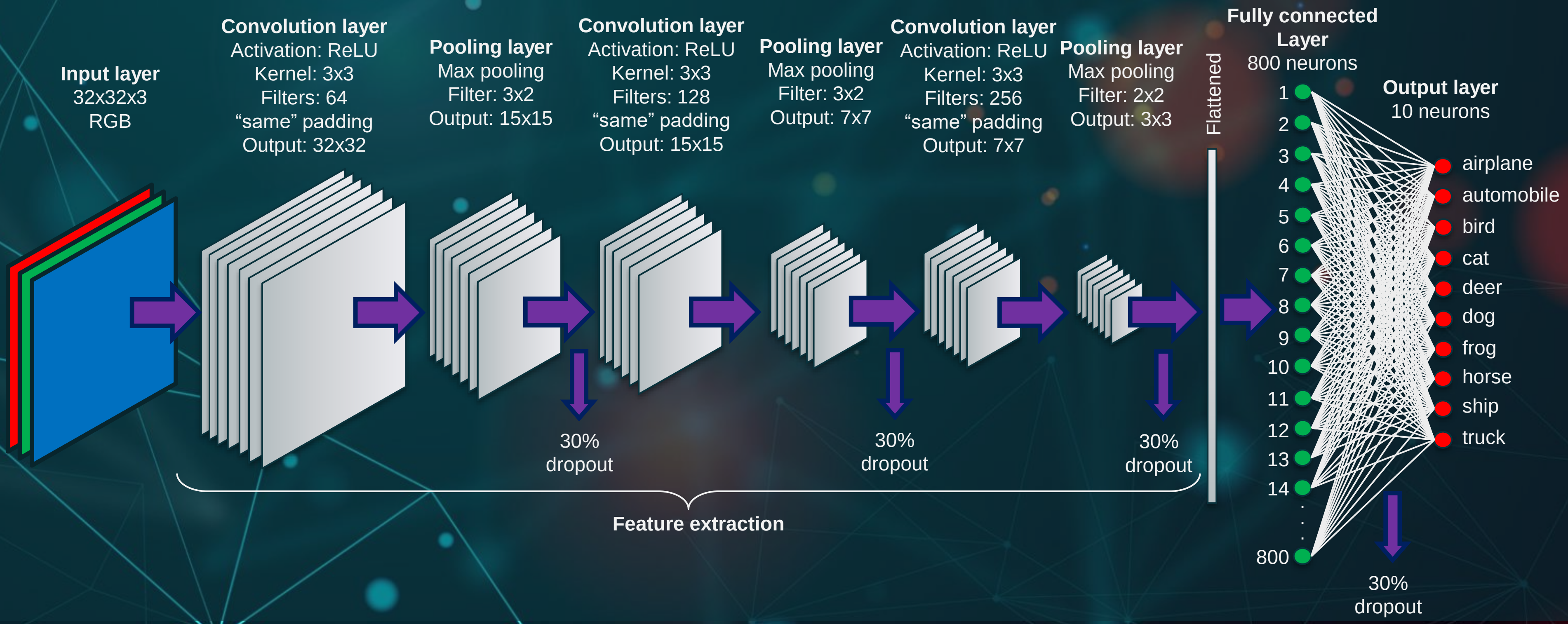
CNN INITIAL DESIGN APPROACH

- Parameters from the “best for far” ANN were initially used:
    - Optimiser: adam
    - Activation: ReLU - the most popular activation function (Zhang et al, 2021), (Sharma et al, 2020)
    - Batch size: 128
    - Loss function: categorical_crossentropy – best for multiclass classification (Brownlee, 2021)
    - Kernel initialiser: default (glorot_uniform)
    - Bias initialiser: default (zeros)
    - Output Activation: softmax - used for multiclass classification(Sharma et al, 2020)
- Number and configuration of layers was experimented with, changing a single parameter at a time adjusting for positive/negative results:
    - Number of convolutional layers, number of filters and kernel size
    - Number of pooling layers and filter size
    - Number of fully connected layers and number of neurons
    - Batch size
    - Stride size
- Epochs determined with early stop function
- Achieved accuracy of 70% but with quite high overfitting

FINALISING THE CNN DESIGN

- Added dropout following CNN seminar:
  - Overfitting decreased significantly with small increase in accuracy
  - Experimented with higher and lower values for optimal dropout value
- Added padding following CNN seminar:
  - Accuracy improved and it opened-up more kernel and pool filter sizes because “same” padding doesn’t reduce the spatial dimensionality of the output of the convolutional layer
- Tested optimiser, activation, kernel initialiser, and bias initialiser from the final ANN model:
  - Optimiser: adamax
    - Model improved so replaced adam with adamax in final CNN model
  - Activation: elu
    - Model diminished so not used
  - Kernel initialiser: HeUniform
    - Model diminished so not used
  - Bias initialiser: TruncatedNormal
    - Model diminished so not used
- After experimenting with square pooling filters, we tried (3,2) which improved. We tried additional rectangles, but none improved on (3,2)
- Final model was tested after converting images to grayscale. The model diminished so RGB was used

CNN FINAL MODEL



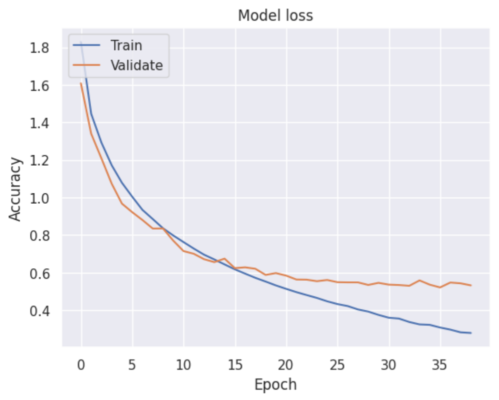
CNN FINAL MODEL

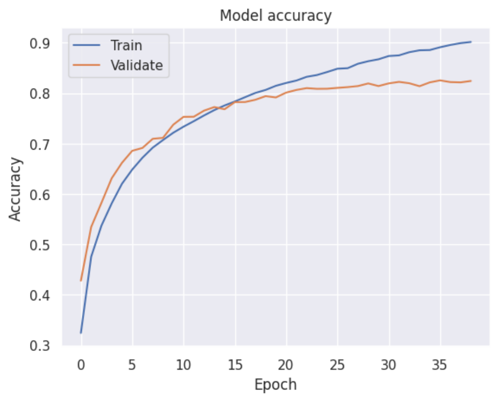
```
model=tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), strides=(1,1), padding='same', activation='relu', input_shape=(32, 32, 3)),
  tf.keras.layers.MaxPooling2D(3,2),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Conv2D(128, (3,3), strides=(1,1), padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(3,2),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Conv2D(256, (3,3), strides=(1,1), padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(800, activation='relu'),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 15, 15, 64)       0
 )

 dropout (Dropout)           (None, 15, 15, 64)        0

 conv2d_1 (Conv2D)           (None, 15, 15, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 128)         0

 conv2d_2 (Conv2D)           (None, 7, 7, 256)         295168

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 256)        0
 2D)

 dropout_2 (Dropout)         (None, 3, 3, 256)         0

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 800)               1844000

 dropout_3 (Dropout)         (None, 800)               0

 dense_1 (Dense)             (None, 10)                8010

=================================================================
Total params: 2,222,826
Trainable params: 2,222,826
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='categorical_crossentropy',
     optimizer='adamax',
     metrics=['acc'])
```

CNN MODEL ACCURACY





Validation set
Loss: 0.5326
Accuracy: 82.44%

Test set
Loss: 0.5533
Accuracy: 82.08%

The confusion matrix of the test set shows good predictions across all classes. Cats performed the worst, which was true of all models tested (both ANN and CNN), with The most common mistake being prediction of a dog.

Actual \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	843	11	31	7	15	2	5	13	52	21
automobile	8	913	6	2	3	3	5	0	16	44
bird	47	0	760	31	61	30	33	21	13	4
cat	19	9	63	628	61	109	43	38	11	19
deer	14	3	52	28	824	13	15	44	7	0
dog	12	1	35	117	43	724	8	46	5	9
frog	5	2	47	40	35	11	844	7	7	2
horse	8	0	16	32	30	26	2	874	6	6
ship	33	19	12	5	3	2	0	1	914	11
truck	21	45	5	7	0	1	5	10	25	881

SUMMARY OF BOTH FINAL MODELS

ANN

Loss function	categorical_crossentropy
Optimiser	adamax
Hidden layers	2 1,000 neurons per layer
Neurons per hidden layer	1,000
Activation function	elu
Kernel initialiser	he_uniform
Bias initialiser	truncated_normal
Dropout	0.5 after every hidden layer
Epochs	108
Validation results	**Loss: 1.2305** **Accuracy: 0.5808 (58.08%)**

CNN

Loss function	categorical_crossentropy
Optimiser	adamax
Convolutional layers	3
Filters per conv layer	64, 128, 256
Kernel size	3x3
Stride	1x1
Passing	same
Activation function	relu
Kernel initialiser	glorot_uniform
Bias initialiser	zeros
Dropout	0.5 after every hidden layer
Fully connected layers	1
Neurons in FC layer	800
Epochs	39
Validation results	**Loss: 0.5326** **Accuracy: 0.8244 (82.44%)**
Test results	**Loss: 0.5533** **Accuracy: 0.8205 (82.05%)**

CONCLUSIONS

- Happy with the final performance of our ANN and CNN models
- We learned a huge amount about building neural networks, including
    - The need to tune hyperparameters; optimiser, activation function, loss function, kernel initialiser, bias initialiser and batch size
    - The impact of increasing neurons and layers
    - The impact of dropout
    - The structure and benefits of convolutional layers and pooling layers in a CNN
    - Increasing epochs increases accuracy through backpropagation, but all models eventually start to overfit, so the early stop function helps to identify the best time to stop
    - How easy it is to build machine learning models with the Keras library in Python
- We worked very well as a team. We were collaborative, divided the work fairly, and communicated regularly.
- Future improvement: identification of best practice hyperparameters.

REFERENCES

Brownlee, J. (2021) How to Choose an Activation Function for Deep Learning. Available from: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ [Accessed 23 June 2023].

Dertat, A. (2017) Applied Deep Learning - Part 4: Convolutional Neural Networks. Available from: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2 [Accessed 23 June 2023].

EDUCBA (2023) Keras Flatten. Available from: https://www.educba.com/keras-flatten/ [Accessed on 8 June 2023].

DJ (2020) Definitive Guide of Activations in Machine Learning. Available from: https://towardsdatascience.com/manual-of-activations-in-deep-learning-30658167ffcb [Accessed 1 July 2023].

Dumane, G. (2020) Introduction to Convolutional Neural Network (CNN) using Tensorflow. Available from: https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83 [Accessed 8 July 2023].

Haan, K. (2023) 24 Top AI Statistics And Trends In 2023. Available from: https://www.forbes.com/advisor/business/ai-statistics/ [Accessed 2 July 2023].

Himanshu, S. Activation Functions : Sigmoid, tanh, ReLU, Leaky ReLU, PReLU, ELU, Threshold ReLU and Softmax basics for Neural Networks and Deep Learning. Available from: https://himanshuxd.medium.com/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e#:~:text=ELU%20have%20been%20shown%20to,slowly%20whereas%20ReLU%20smooths%20sharply [Accessed 1 July 2023].

IBM (N.D.) What is artificial intelligence? Available from: https://www.ibm.com/topics/artificial-intelligence [Accessed 2 July 2023].

Keras (2020a) The Sequential model. Available from: https://keras.io/guides/sequential_model/ [Accessed 8 July 2023].

Keras (2020b) Layer weight regularizers. Available from: https://keras.io/api/layers/regularizers/ [Access 8 July 2023].

Keras (2022) Training-related part of keras engine. Available from: https://github.com/keras-team/keras/blob/68dc181a5e34d1f20edabe531176b3bfb50001f9/keras/engine/training.py#L375 [Accessed 9 July 2023].

Keras (N.D.) CIFAR10 small images classification dataset. Available from: https://keras.io/api/datasets/cifar10/ [Accessed 23 June].

REFERENCES

Keras (N.D.) Accuracy metrics. Available from: https://keras.io/api/metrics/accuracy_metrics/#sparsecategoricalaccuracy-class [Access 9 July 2023].

Krizhevsky, A. (2009) *Learning Multiple Layers of Features from Tiny Images.* Available from: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf [Accessed 23 June 2023].

Pramoditha, R. (2022) Size, Epochs and Training Steps in a Neural Network. Available from: https://medium.com/data-science-365/all-you-need-to-know-about-batch-size-epochs-and-training-steps-in-a-neural-network-f592e12cdb0a [Accessed 9 July 2023]

ProjectPro (2022) What is the use of activation functions in keras ? Available from: https://www.projectpro.io/recipes/what-is-use-of-activation-functions-keras [Accessed 8 July 2023].

Russell, S. & Norvig, P. (2021) *Artificial Intelligence: A Modern Approach.* 4th ed. Harlow: Pearson Education Limited.

Sharma, S., Sharma, S. & Athaiya, A. (2020) Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 4(12): 310-316.

Srivastana, N., Hointon, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15(1): 1929-1958.

Sultana, F., Sufian, A. & Dutta, P. (2018) 'Advancements in image classification using convolutional neural network', *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. Kolkata, India, 22-23 November. IEEE. 122-129

Wang, Z.J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., Kahng, M. & Chau, D.H.P. (2020) CNN explainer: learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(2): 1396-1406.

World Economic Forum (2023) *The Future of Jobs Report 2023*. Available from: https://www3.weforum.org/docs/WEF_Future_of_Jobs_2023.pdf [Accessed 2 July 2023].

Zhang, X., Chang, D., Qi, W. & Zhan, Z. (2021) A Study on different functionalities and performances among different activation functions across different ANNs for image classification. *Journal of Physics: Conference Series,* Vol. 1732(1): 1-6.