University of Essex Online

Development Individual Project: Presentation Developing a Neural Network with the CIFAR-10 image dataset

William Bolton

w.s.bolton@leeds.ac.uk

24th May 2024

AI and Object Recognition

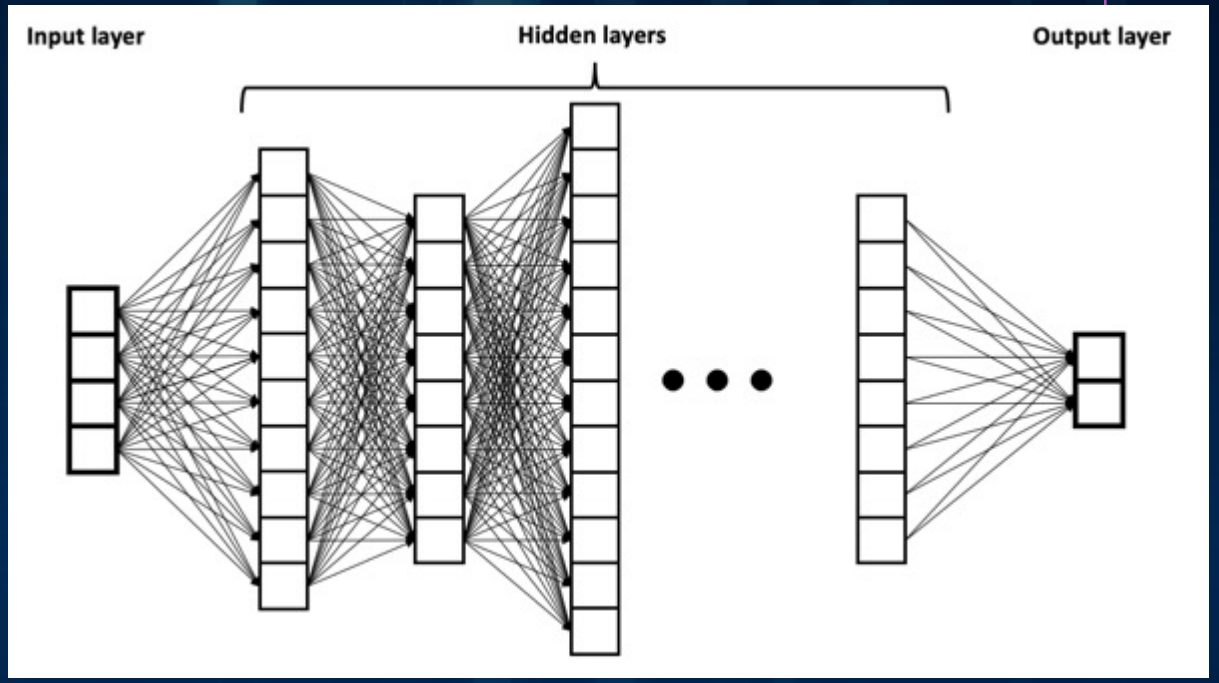
Autonomous Vehicles

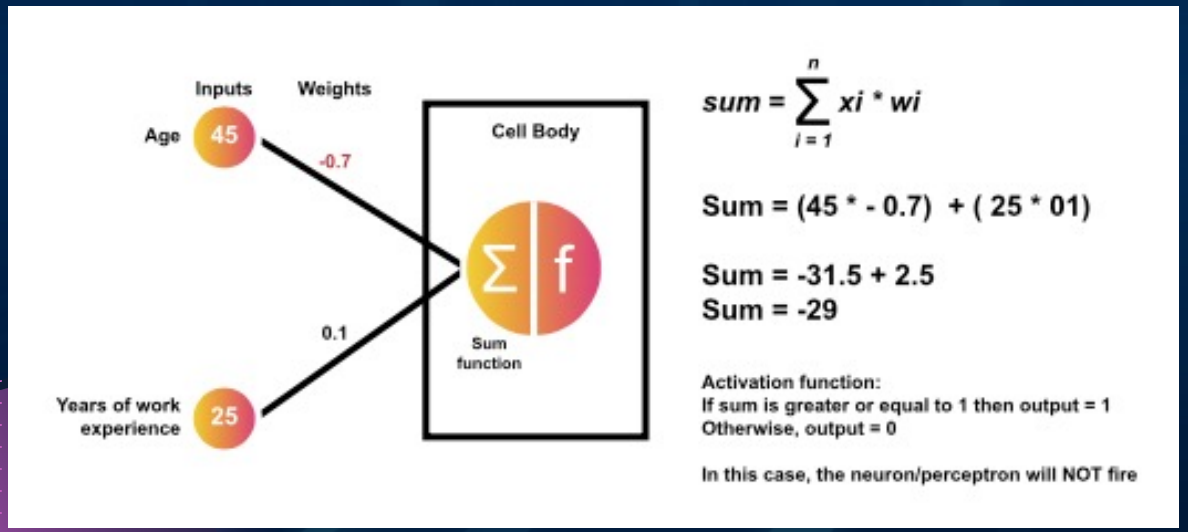
Diagnostic medical imaging

Inventory management

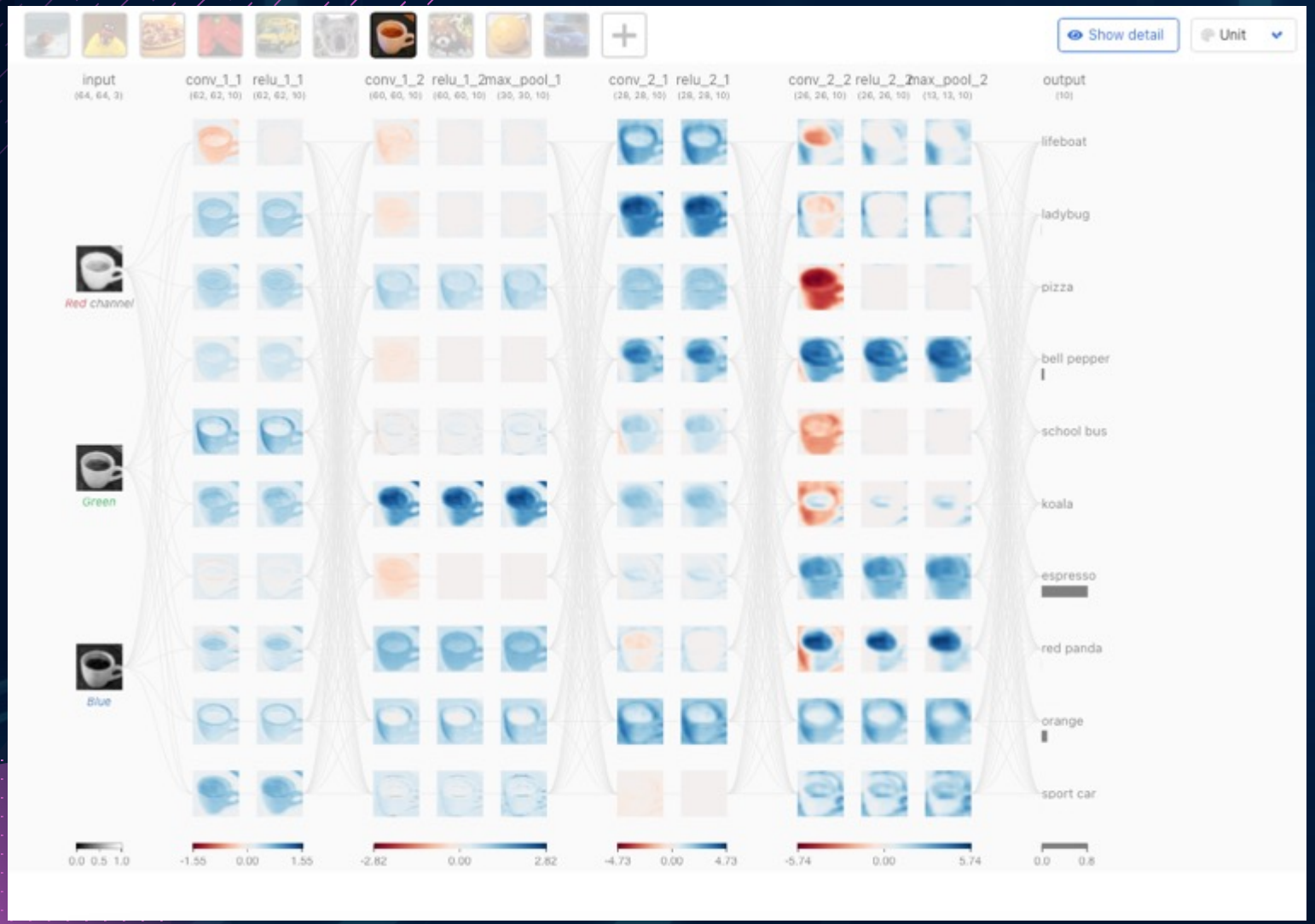
Object recognition is a critical function of many AI systems that typically required human intelligence.

Neural Networks: A backbone of AI





Architecture of the Artificial Neural Network



- Standard feed-forward convolutional neural network
- Sequential model with Conv2D, MaxPooling2D, Flatten, Dense layers
- Use ReLU activation for hidden layers, softmax for output layer
- Before I could test this design, I prepared and partitioned the data

Imports, setup, and preparing the dataset

```
1   import os
2   import pickle
3   import tensorflow as tf
4   from tensorflow.keras.datasets import cifar10
5   from tensorflow.keras.models import Sequential
6   from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
7   from tensorflow.keras.utils import to_categorical
8   import numpy as np
9   import matplotlib.pyplot as plt
10  from sklearn.metrics import precision_score, recall_score, f1_score
```

```
41
42   # Normalize the data
43   train_images, test_images = train_images / 255.0, test_images / 255.0
44
```

Partitioning the validation set from training data and its importance

```python
# Ensures representative splitting
def representative_split(X, y, test_size=0.2, random_state=42):
    np.random.seed(random_state)
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    split = int(X.shape[0] * (1 - test_size))
    train_idx, test_idx = indices[:split], indices[split:]
    return X[train_idx], X[test_idx], y[train_idx], y[test_idx]

# Partition the training data to create a validation set using representative split
train_images, validation_images, train_labels, validation_labels = representative_split(train_images, train_labels, test_size=0.1)
```

```python
# Data specifics (metadata)
print(f"Training samples: {train_images.shape[0]}")
print(f"Validation samples: {validation_images.shape[0]}")
print(f"Test samples: {test_images.shape[0]}")
```

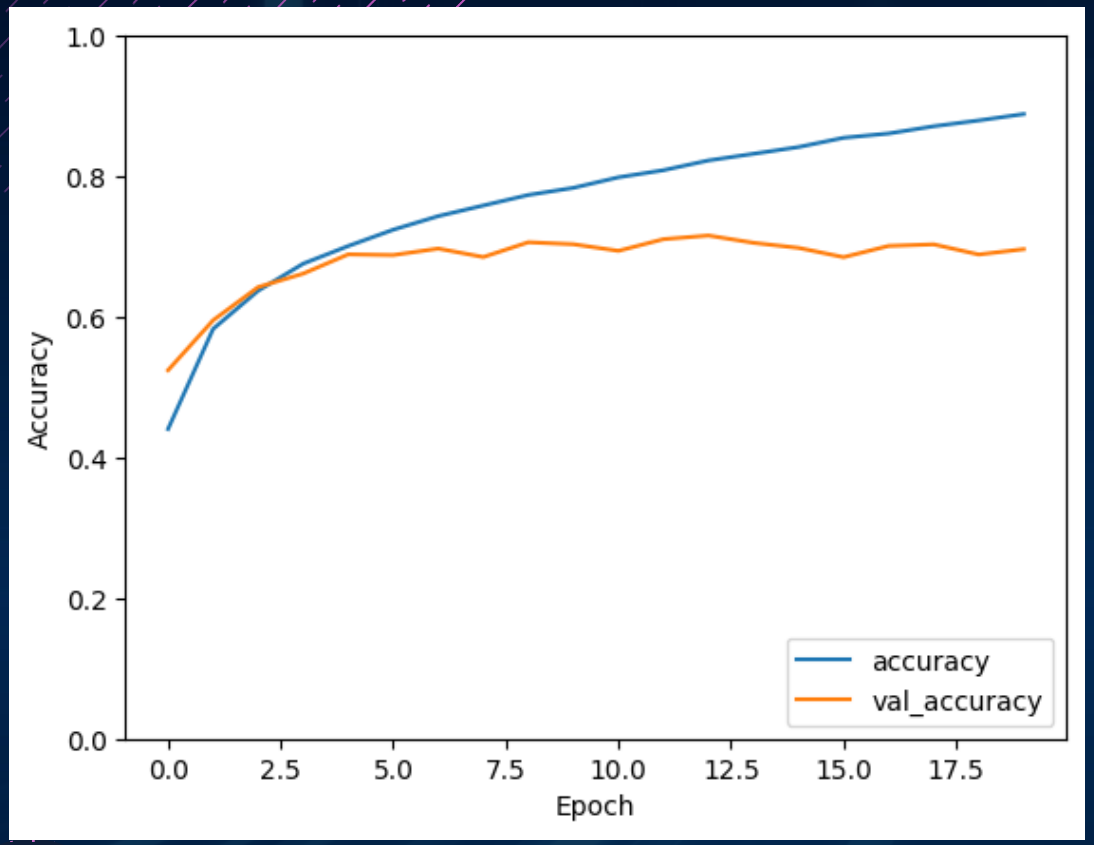
Training samples: 45000
Validation samples: 5000
Test samples: 10000

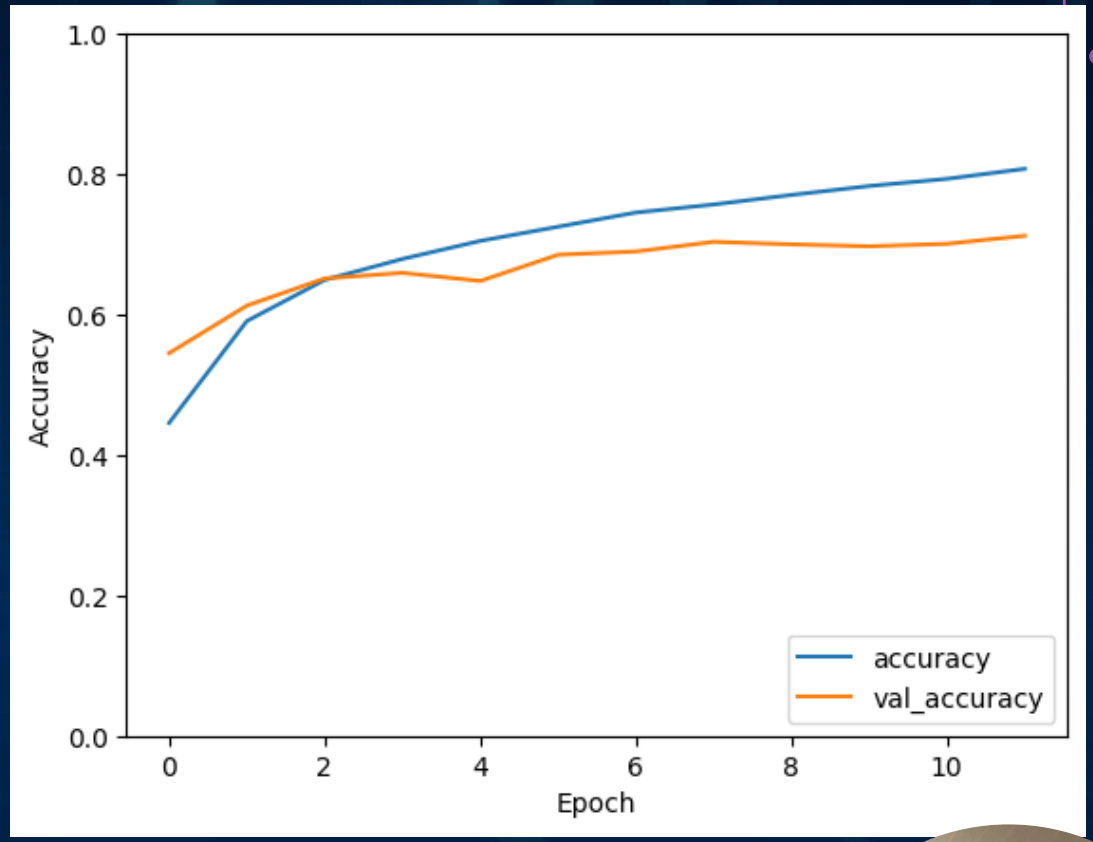
Activation and Loss Function

$$\mathrm{ReLU}(x) = \max(0, x)$$

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i = \frac{1}{N}\sum_{i=1}^{N} -\log(\hat{y}_{i,y_i})$$

Epochs in the modelling process





Neural Network's Design Strategy

```python
# Define the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
```

```python
# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(train_images, train_labels, epochs=12,
                    validation_data=(validation_images, validation_labels))

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")
```

Reflecting and conclusions on learnings acquired

```
Epoch 1/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 32s 22ms/step - accuracy: 0.3408 - loss: 1.7885 - val_accuracy: 0.5104 - val_loss: 1.3531
Epoch 2/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 33s 23ms/step - accuracy: 0.5579 - loss: 1.2339 - val_accuracy: 0.6192 - val_loss: 1.0882
Epoch 3/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 34s 24ms/step - accuracy: 0.6261 - loss: 1.0543 - val_accuracy: 0.6406 - val_loss: 1.0188
Epoch 4/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 33s 24ms/step - accuracy: 0.6670 - loss: 0.9489 - val_accuracy: 0.6514 - val_loss: 1.0065
Epoch 5/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 34s 24ms/step - accuracy: 0.6938 - loss: 0.8753 - val_accuracy: 0.6794 - val_loss: 0.9024
Epoch 6/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 34s 24ms/step - accuracy: 0.7166 - loss: 0.8070 - val_accuracy: 0.6838 - val_loss: 0.9094
Epoch 7/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 33s 24ms/step - accuracy: 0.7347 - loss: 0.7534 - val_accuracy: 0.6812 - val_loss: 0.9259
Epoch 8/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 34s 24ms/step - accuracy: 0.7484 - loss: 0.7126 - val_accuracy: 0.7008 - val_loss: 0.8817
Epoch 9/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 33s 23ms/step - accuracy: 0.7608 - loss: 0.6768 - val_accuracy: 0.6938 - val_loss: 0.9330
Epoch 10/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 32s 23ms/step - accuracy: 0.7803 - loss: 0.6331 - val_accuracy: 0.6956 - val_loss: 0.9164
Epoch 11/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 32s 23ms/step - accuracy: 0.7843 - loss: 0.6081 - val_accuracy: 0.7088 - val_loss: 0.8630
Epoch 12/12
1407/1407 ━━━━━━━━━━━━━━━━━━━━ 32s 22ms/step - accuracy: 0.7958 - loss: 0.5683 - val_accuracy: 0.7074 - val_loss: 0.8699
313/313 - 2s - 8ms/step - accuracy: 0.7061 - loss: 0.8953
Test accuracy: 0.7060999870300293
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step
Test precision: 0.7108038470261907
Test recall: 0.7061
Test F1 score: 0.7060825735915871
```

References

CNN Explainer: Learn Convolutional Neural Network Webpage by Wang, J; Turko: R; Shaikh, O; Park, H; Das, N; Hohman, F; Kahng, M; Chau, P. Online at: https://poloclub.github.io/cnn-explainer/ Accessed 24.05.2024.

Kubat, M., 2017. An introduction to machine learning. Springer.

Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

File: Example of a deep neural network.png, From Wikimedia Commons, the free media repository. Online at: https://commons.wikimedia.org/wiki/File:Example_of_a_deep_neural_network.png accessed 24.05.2024.

The Essential Guide to Neural Network Architectures, by Pragati Baheti (Microsoft), V7. Online at: https://www.v7labs.com/blog/neural-network-architectures-guide Accessed 24.05.2024.

Techniques for training large neural networks, OpenAI. Online at: https://openai.com/index/techniques-for-training-large-neural-networks/ Accessed 24.05.2024.

Will Cukierski. (2013). CIFAR-10 - Object Recognition in Images. Kaggle. https://kaggle.com/competitions/cifar-10

University of Essex Online

Development Individual Project: Presentation Developing a Neural Network with the CIFAR-10 image dataset

William Bolton

w.s.bolton@leeds.ac.uk

24th May 2024