Convolutional Neural Networks

Cats See in Patterns

Prompts

- I have two sets of pictures of cats and dogs. Could you make CNN tell them apart?
- Could you give a TensorFlow/Keras version?

Strategy

- Prepare the data
- Build a neural network, compile the model, train it, and evaluate
- Visualize training

Code & Results

1. Prepare the data

This code loads the dataset and normalizes the data.

```
import tensorflow as tf
from tensorflow.keras import layers, models

# === 1. Load dataset ===
train_ds = tf.keras.utils.image_dataset_from_directory(
    "data/train",
    image_size=(64, 64),  # smaller size = less memory
    batch_size=16
)

val_ds = tf.keras.utils.image_dataset_from_directory(
    "data/val",
    image_size=(64, 64),
    batch_size=16
)

# === 2. Normalize ===
train_ds = train_ds.map(lambda x, y: (x / 255.0, y))
val_ds = val_ds.map(lambda x, y: (x / 255.0, y))
```

≡ kaggle

kaggle.com/datasets/

Cats-And-Dogs-Mini-Dataset

```
data/
├── train/
│   ├── cats/   (≈400 images)
│   └── dogs/   (≈400 images)
├── val/
│   ├── cats/   (≈100 images)
│   └── dogs/   (≈100 images)
```

2. Build and evaluate the Model

This code builds a CNN, compiles and trains the model, and evaluates loss and accuracy.

```
# === 3. Small CNN ===
model = models.Sequential([
    layers.Conv2D(16, (3,3), activation='relu', input_shape=(64,64,3)),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(32, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

# === 4. Compile ===
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# === 5. Train ===
history = model.fit(train_ds, validation_data=val_ds, epochs=25)

# === 6. Save model ===
model.save("cat_dog_small_cnn.h5")

# === 7. Evaluate ===
loss, acc = model.evaluate(val_ds)
print(f"Validation accuracy: {acc:.2f}")
```

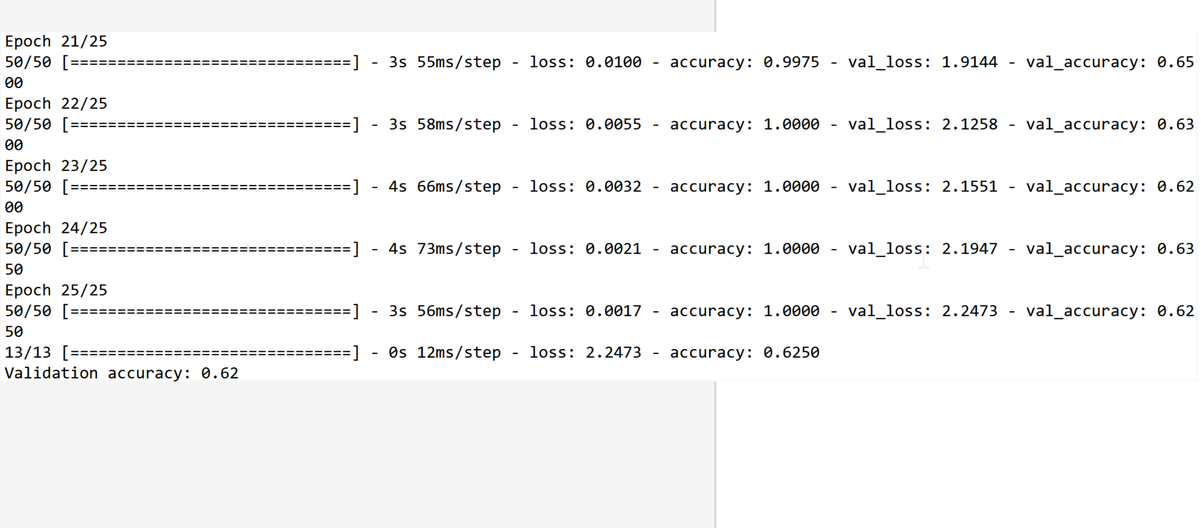
```
Epoch 21/25
50/50 [==============================] - 3s 55ms/step - loss: 0.0100 - accuracy: 0.9975 - val_loss: 1.9144 - val_accuracy: 0.65
00
Epoch 22/25
50/50 [==============================] - 3s 58ms/step - loss: 0.0055 - accuracy: 1.0000 - val_loss: 2.1258 - val_accuracy: 0.63
00
Epoch 23/25
50/50 [==============================] - 4s 66ms/step - loss: 0.0032 - accuracy: 1.0000 - val_loss: 2.1551 - val_accuracy: 0.62
00
Epoch 24/25
50/50 [==============================] - 4s 73ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 2.1947 - val_accuracy: 0.63
50
Epoch 25/25
50/50 [==============================] - 3s 56ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 2.2473 - val_accuracy: 0.62
50
13/13 [==============================] - 0s 12ms/step - loss: 2.2473 - accuracy: 0.6250
Validation accuracy: 0.62
```

This code visualizes training.

```
import matplotlib.pyplot as plt

# ===== Plot training results =====
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.legend(); plt.title('Loss')

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='train_acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.legend(); plt.title('Accuracy')
plt.show()
```



Made by: @katerynakononova

Website: https://katerynakononova.github.io/meowlearning/

Book: https://www.amazon.com/dp/BOCW9SFYXF