

```
# Импорт библиотек
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Параметры
params = {
    "input_shape": (224, 224, 3),
    "num_classes": 10,
    "batch_size": 32,
    "epochs": 50,
    "dropout_conv": 0.25,
    "dropout_dense": 0.5,
    "learning_rate": 0.001
}

# Загрузка данных
train_data = np.load('/content/drive/MyDrive/train_small.npz')
test_data = np.load('/content/drive/MyDrive/test_small.npz')

# Предобработка данных
def preprocess_data(data, labels):
    data = data.astype(np.float32) / 255.0
    return data, labels

X_train, y_train = preprocess_data(train_data['data'], train_data['labels'])
X_test, y_test = preprocess_data(test_data['data'], test_data['labels'])

# Аугментация данных
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)
datagen.fit(X_train)

# Создание модели
def create_model(input_shape, num_classes, dropout_conv, dropout_dense):
    model = Sequential([
        Input(shape=input_shape),
        Conv2D(32, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Dropout(dropout_conv),
```

```

        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Dropout(dropout_conv),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Dropout(dropout_conv),
        Flatten(),
        Dense(256, activation='relu'),
        Dropout(dropout_dense),
        Dense(num_classes, activation='softmax')
    ])
    return model

```

```

model = create_model(params["input_shape"], params["num_classes"], params["dropout_conv"],
optimizer = tf.keras.optimizers.Adam(learning_rate=params["learning_rate"]))
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

```

Обучение

```

callbacks = [
    tf.keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True),
    tf.keras.callbacks.ModelCheckpoint("best_model.keras", save_best_only=True)
]

```

```

history = model.fit(
    datagen.flow(X_train, y_train, batch_size=params["batch_size"]),
    epochs=params["epochs"],
    validation_data=(X_test, y_test),
    callbacks=callbacks
)

```

Оценка

```

loss, accuracy = model.evaluate(X_test, y_test)
print(f"Точность улучшенной модели CNN: {accuracy:.2f}")

```

Отчет и матрица ошибок

```

y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
print("Отчет классификации:")
print(classification_report(y_test, y_pred_classes, zero_division=0))
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_classes, cmap="Blues")
plt.title("Confusion Matrix")
plt.show()

```

Визуализация обучения

```

def plot_history(history):
    plt.figure(figsize=(12, 6))
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.legend()


```

```
plt.legend()
plt.title("Training and Validation Metrics")
plt.show()
```


plot_history(history)

```
... Epoch 1/50
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_a
self._warn_if_super_not_called()
225/225 ██████████ 1133s 5s/step - accuracy: 0.1139 - loss: 2.9320 - val_ac
Epoch 2/50
225/225 ██████████ 1174s 5s/step - accuracy: 0.2366 - loss: 2.0020 - val_ac
Epoch 3/50
225/225 ██████████ 1146s 5s/step - accuracy: 0.3630 - loss: 1.6177 - val_ac
Epoch 4/50
225/225 ██████████ 1140s 5s/step - accuracy: 0.4834 - loss: 1.3176 - val_ac
Epoch 5/50
225/225 ██████████ 1117s 5s/step - accuracy: 0.5242 - loss: 1.1816 - val_ac
Epoch 6/50
225/225 ██████████ 1152s 5s/step - accuracy: 0.5679 - loss: 1.0687 - val_ac
Epoch 7/50
225/225 ██████████ 1123s 5s/step - accuracy: 0.5792 - loss: 1.0303 - val_ac
Epoch 8/50
225/225 ██████████ 1102s 5s/step - accuracy: 0.6199 - loss: 0.9529 - val_ac
Epoch 9/50
225/225 ██████████ 1115s 5s/step - accuracy: 0.6169 - loss: 0.9564 - val_ac
Epoch 10/50
225/225 ██████████ 1112s 5s/step - accuracy: 0.6303 - loss: 0.9196 - val_ac
Epoch 11/50
225/225 ██████████ 1099s 5s/step - accuracy: 0.6522 - loss: 0.8886 - val_ac
Epoch 12/50
225/225 ██████████ 1109s 5s/step - accuracy: 0.6573 - loss: 0.8627 - val_ac
Epoch 13/50
225/225 ██████████ 1163s 5s/step - accuracy: 0.6707 - loss: 0.8392 - val_ac
Epoch 14/50
225/225 ██████████ 1125s 5s/step - accuracy: 0.6899 - loss: 0.8079 - val_ac
Epoch 15/50
225/225 ██████████ 1088s 5s/step - accuracy: 0.7057 - loss: 0.7838 - val_ac
Epoch 16/50
225/225 ██████████ 1111s 5s/step - accuracy: 0.7002 - loss: 0.7766 - val_ac
Epoch 17/50
225/225 ██████████ 1083s 5s/step - accuracy: 0.7177 - loss: 0.7467 - val_ac
Epoch 18/50
225/225 ██████████ 1132s 5s/step - accuracy: 0.7084 - loss: 0.7776 - val_ac
Epoch 19/50
225/225 ██████████ 1137s 5s/step - accuracy: 0.7418 - loss: 0.7023 - val_ac
Epoch 20/50
225/225 ██████████ 1109s 5s/step - accuracy: 0.7372 - loss: 0.7246 - val_ac
Epoch 21/50
225/225 ██████████ 1094s 5s/step - accuracy: 0.7407 - loss: 0.6801 - val_ac
Epoch 22/50
225/225 ██████████ 1108s 5s/step - accuracy: 0.7592 - loss: 0.6629 - val_ac
Epoch 23/50
225/225 ██████████ 1117s 5s/step - accuracy: 0.7465 - loss: 0.6808 - val_ac
```

Эпока 24/50

225/225  **1158s** 5s/step - accuracy: 0.7731 - loss: 0.6229 - val_ac

Epoch 25/50

151/225  **5:30** 4s/step - accuracy: 0.7781 - loss: 0.6186