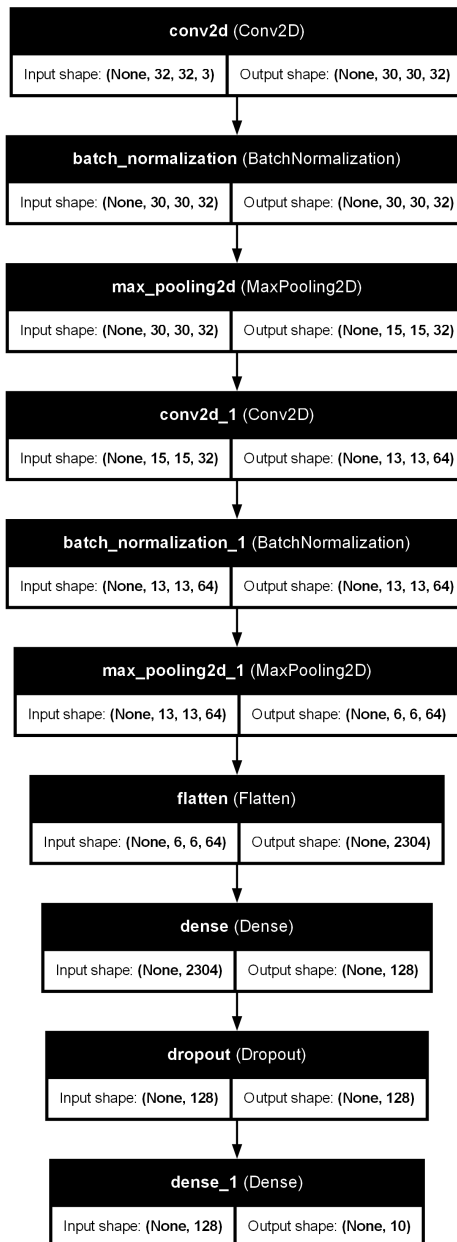


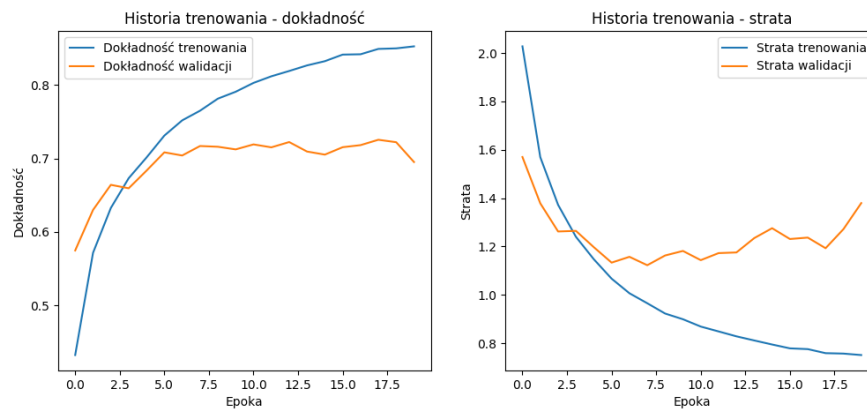
Projekt 5

Precyzja na danych testowych: 0.70

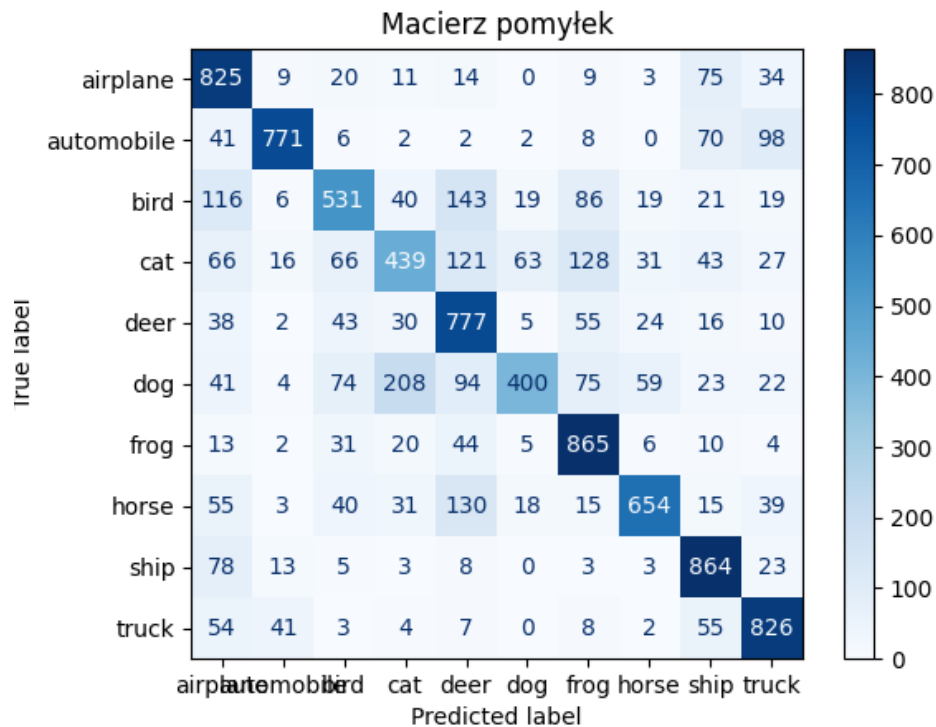
Architektura sieci



Historia trenowania



Macierz pomyłek



Kod źródłowy

```
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.utils import plot_model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.initializers import HeNormal
from tensorflow.keras.optimizers import Adam
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```

# === LOAD DATA ===
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# === MODEL ===
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', kernel_initializer=HeNormal(), kernel_regularizer=l2(0.001), input_shape=(32, 32, 3)),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu', kernel_initializer=HeNormal(), kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    Flatten(),
    Dense(128, activation='relu', kernel_initializer=HeNormal(), kernel_regularizer=l2(0.001)),
    Dropout(0.4),
    Dense(10, activation='softmax')
])
model.compile(optimizer=Adam(learning_rate=0.0005),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=20, batch_size=64, validation_data=(x_test, y_test))

# === SAVE MODEL AND PLOTS ===
model.save('cifar10_model.keras')
plot_model(model, to_file='model_architecture.png', show_shapes=True, show_layer_names=True)

# Confusion matrix
y_pred_prob = model.predict(x_test)
y_pred = np.argmax(y_pred_prob, axis=1)
y_true = np.argmax(y_test, axis=1)
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap=plt.cm.Blues)
plt.title("Macierz pomyłek")
plt.savefig("confusion_matrix.png")
plt.show()

# Training history
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Dokładność trenowania')
plt.plot(history.history['val_accuracy'], label='Dokładność walidacji')
plt.title('Historia trenowania - dokładność')
plt.xlabel('Epoka')
plt.ylabel('Dokładność')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Strata trenowania')
plt.plot(history.history['val_loss'], label='Strata walidacji')
plt.title('Historia trenowania - strata')
plt.xlabel('Epoka')
plt.ylabel('Strata')
plt.legend()

plt.savefig("training_history.png")
plt.show()

test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f"Precyzja na danych testowych: {test_accuracy:.2f}")

```

```
# === PDF REPORT ===
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image, Preformatted
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib import colors
from reportlab.lib.units import inch
from PIL import Image as PILImage

# ██████████ ████████████████████ ████████████████████ ███ ██████████ ██████████

def get_resized_image(path, max_width, max_height):
    pil_img = PILImage.open(path)
    aspect = pil_img.height / pil_img.width
    width = max_width
    height = width * aspect

    if height > max_height:
        height = max_height
        width = height / aspect

    return Image(path, width=width, height=height)

def create_pdf_report(filename, test_accuracy, source_code):
    doc = SimpleDocTemplate(filename, pagesize=A4)
    styles = getSampleStyleSheet()
    story = []

    story.append(Paragraph("Projekt 5", styles['Title']))
    story.append(Spacer(1, 12))

    story.append(Paragraph(f"<b>Precyzja na danych testowych:</b> {test_accuracy:.2f}", styles['Normal']))
    story.append(Spacer(1, 12))

    max_image_width = 5.5 * inch
    max_image_height = 6.5 * inch

    for img_path, caption in [
        ("model_architecture.png", "Architektura sieci"),
        ("training_history.png", "Historia trenowania"),
        ("confusion_matrix.png", "Macierz pomyłek"),
    ]:
        story.append(Paragraph(caption, styles['Heading2']))
        story.append(Spacer(1, 6))
        img = get_resized_image(img_path, max_image_width, max_image_height)
        story.append(img)
        story.append(Spacer(1, 12))

    code_style = ParagraphStyle(
        'Code',
        fontName='Courier',
        fontSize=8,
        leading=10,
        backColor=colors.whitesmoke,
        borderWidth=0.5,
        borderColor=colors.gray,
        borderPadding=5,
        leftIndent=10,
        rightIndent=10,
        spaceAfter=10
    )
    story.append(Paragraph("Kod źródłowy", styles['Heading2']))
    story.append(Preformatted(source_code, code_style))

    doc.build(story)

# ██████████ ████████████████████ ██████████

with open("Project.py", "r", encoding="utf-8") as f:
    source_code = f.read()
```

```
# ■■■■■■■■ PDF ■■■■■■■■  
create_pdf_report("cifar10_report.pdf", test_accuracy, source_code)
```