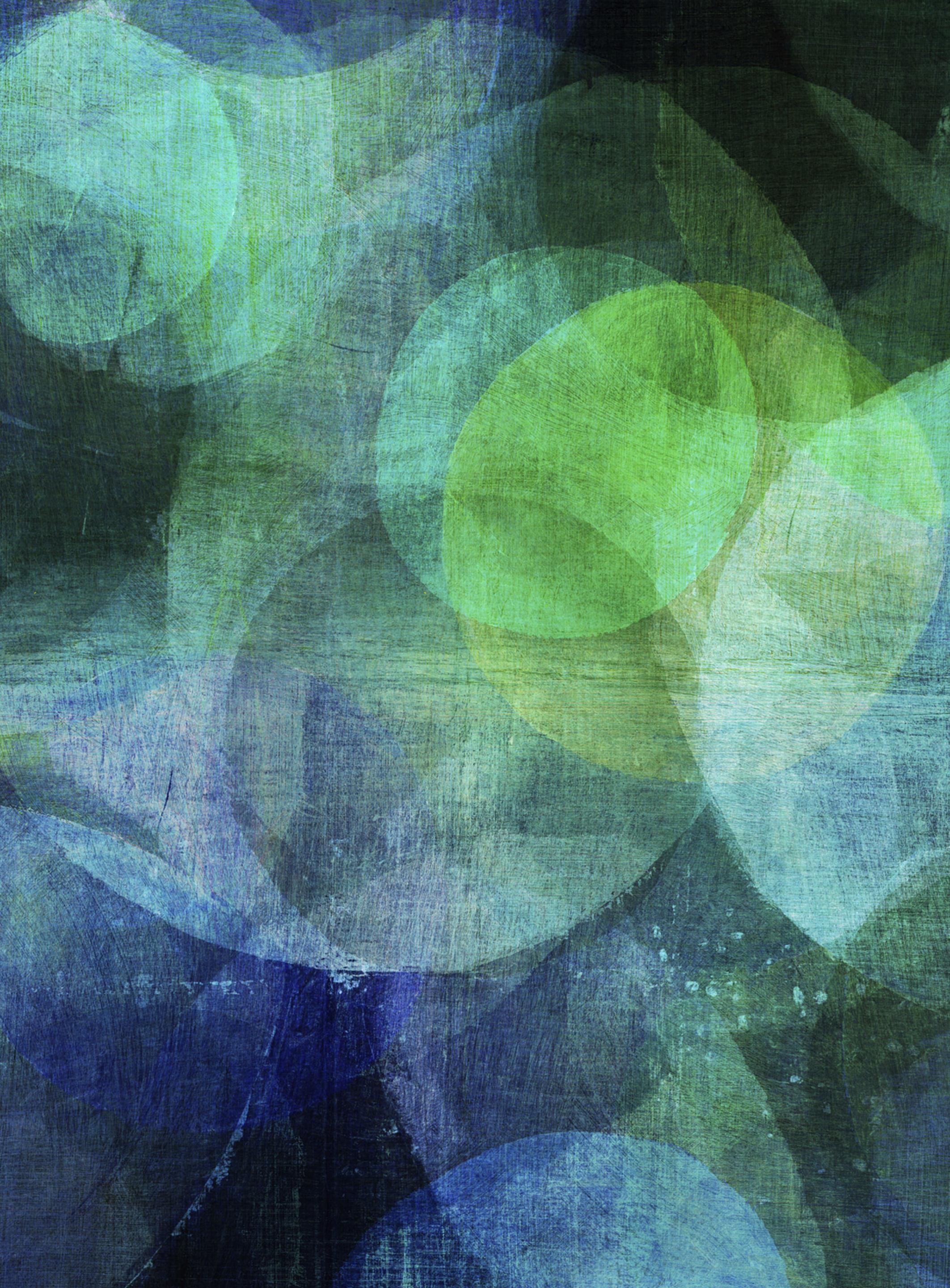


# SZTUCZNA INTELIGENCJA Z LUDZKĄ TWARZĄ

---

*Trenowanie modelu do rozpoznawania  
emocji*



# **BAZA DANYCH**



# BAZA DANYCH: FACE EXPRESSION RECOGNITION DATASET

---

## 📁 Źródło danych:

Projekt opiera się na otwartym zbiorze danych **Face Expression Recognition Dataset** dostępnym na platformie **Kaggle**:

🔗 <https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset>

## 🎯 Cel zbioru:

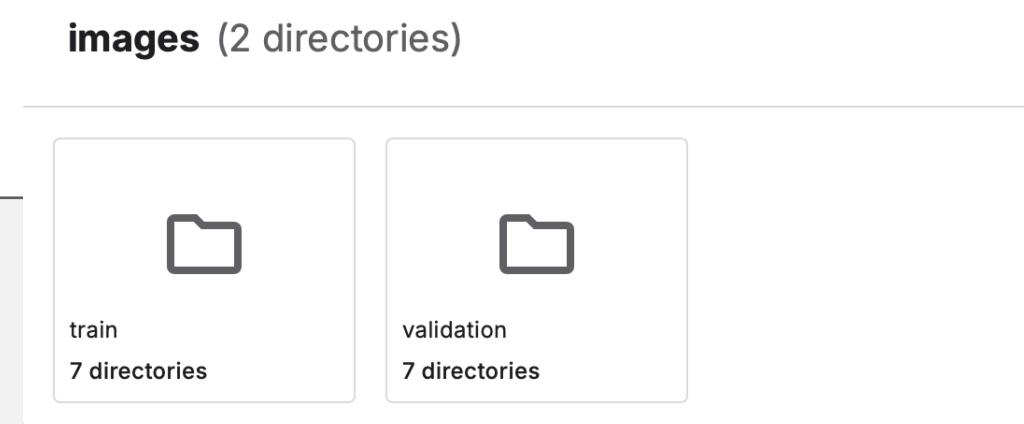
Zestaw danych został przygotowany z myślą o trenowaniu modeli **rozpoznawania emocji na podstawie mimiki twarzy**. Jest to jedno z najpopularniejszych źródeł wykorzystywanych w zadaniach z zakresu:

- **Computer Vision** (komputerowe rozpoznawanie obrazów),
- **Uczenia głębokiego (Deep Learning)**,
- **Sztucznej inteligencji w analizie emocji**.



# BAZA DANYCH: FACE EXPRESSION RECOGNITION DATASET

## 📦 Charakterystyka zbioru:

Atrybut	Wartość
Liczba klas (emocji)	7 
Klasy emocji	Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral
Typ obrazów	Zdjęcia twarzy w skali szarości
Rozmiar obrazów	Zmieniony na 96x96 pikseli dla kompatybilności z modelem
Źródło	Zdjęcia twarzy zebrane w warunkach naturalnych, zróżnicowanych pod względem oświetlenia, kąta, mimiki
Struktura katalogów	Dane podzielone na foldery train/ oraz validation/, każdy z podfolderami odpowiadającymi emocjom

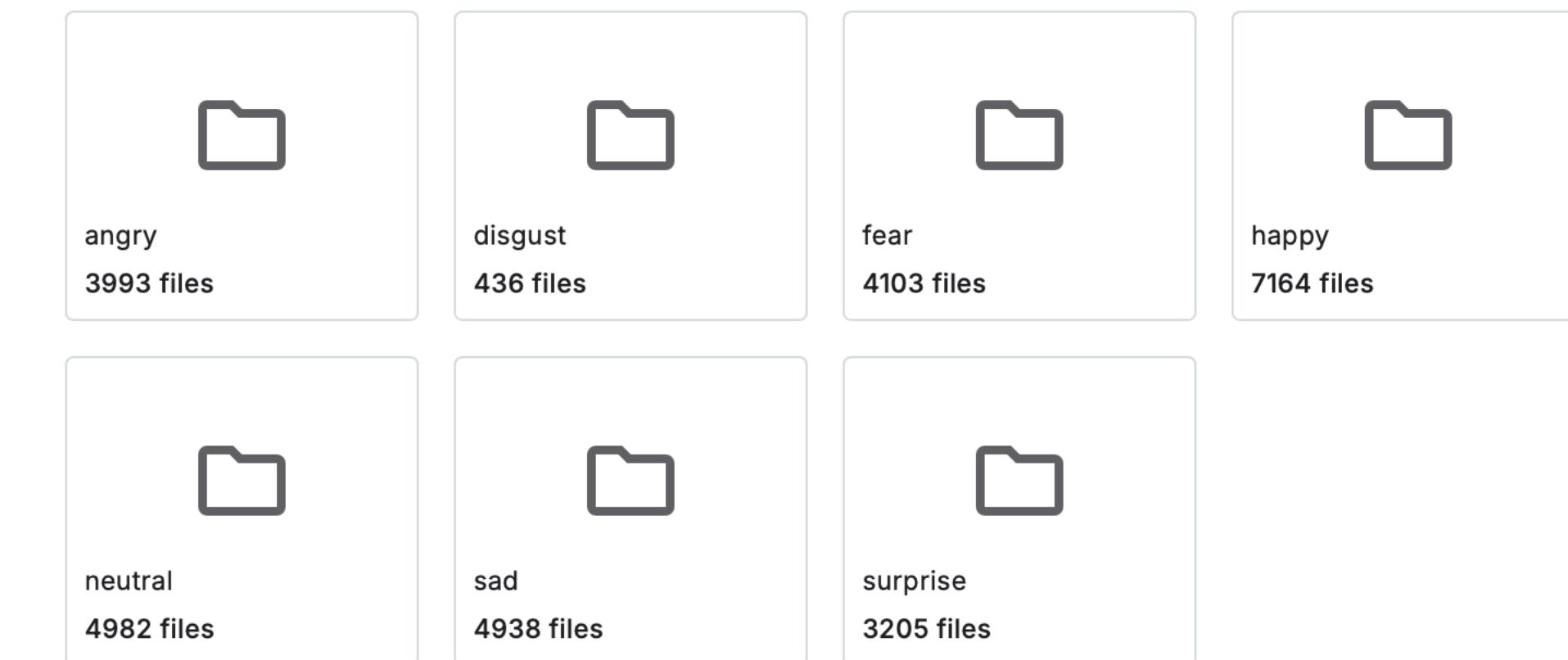


# BAZA DANYCH: FACE EXPRESSION RECOGNITION DATASET

## Rozkład klas w zbiorze treningowym:

Emocja	Liczba próbek
怒怒 Angry	3993
恶心 Disgust	436
惊恐 Fear	4103
高兴 Happy	7164
悲伤 Sad	4982
惊讶 Surprise	4938
中性 Neutral	3205

train (7 directories)



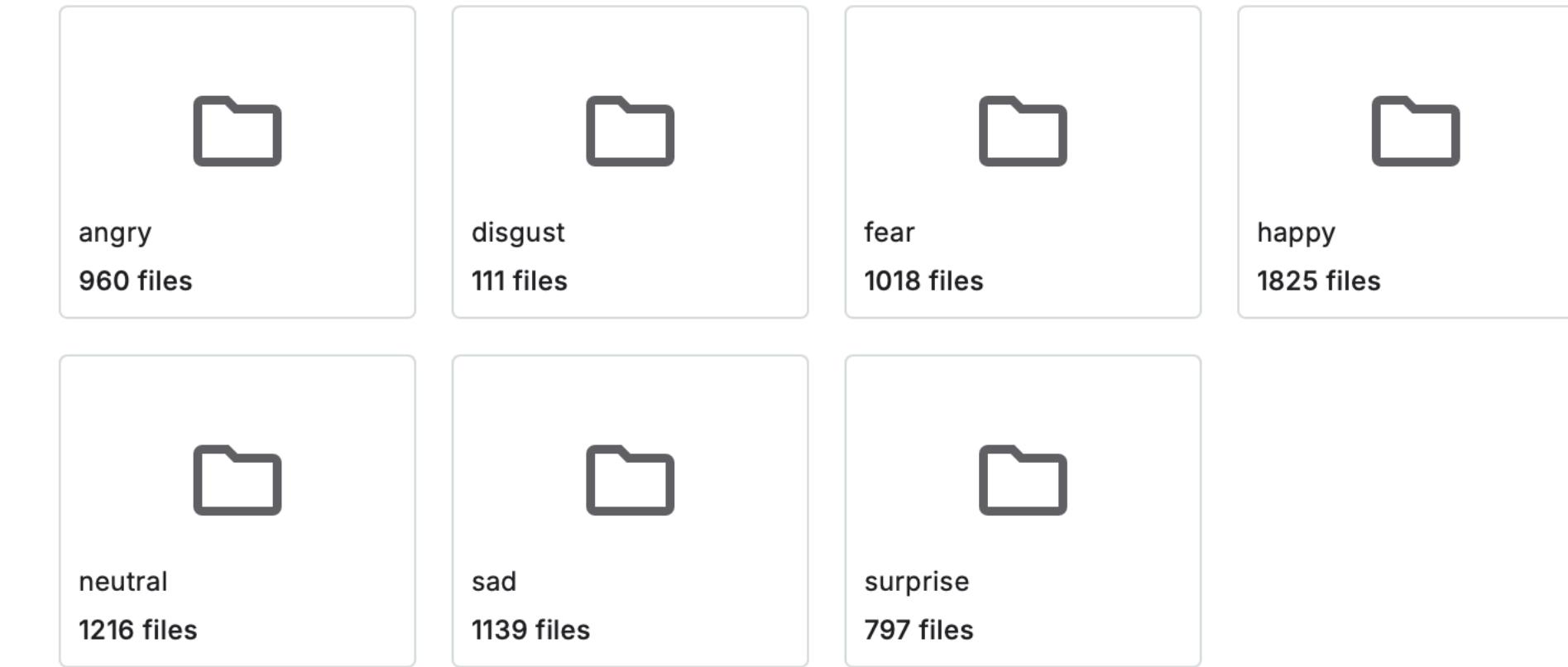


# BAZA DANYCH: FACE EXPRESSION RECOGNITION DATASET

## Rozkład klas w zbiorze testowym:

Emocja	Liczba próbek
怒怒 Angry	960
恶心恶心 Disgust	111
惊恐惊恐 Fear	1018
高兴高兴 Happy	1825
悲伤悲伤 Sad	1216
惊讶惊讶 Surprise	1139
中性中性 Neutral	797

validation (7 directories)



# BALANS KLAS (ANG. CLASS WEIGHTING)

Niektóre klasy (np. emocje) są **znacznie rzadziej reprezentowane** w zbiorze treningowym niż inne. Bez balansu model byłby skłonny faworyzować klasy dominujące.

Dzięki class\_weight, model:

- nadaje **większą wagę rzadkim klasom,**
- **mniej karze** za pomyłki w częstych klasach,
- uczy się **sprawiedliwie**, niezależnie od liczby przykładów.

```
train_counts = [7164, 436, 3993, 4982, 4103, 4938, 3205]

class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.arange(len(class_labels)),
    y=np.repeat(np.arange(len(class_labels)), train_counts)
)
class_weights_dict = {i: w for i, w in enumerate(class_weights)}
```

## Przykład:

Jeśli masz:

- klasę A z 7000 przykładów,
- klasę B z 400 przykładami,

to class\_weight automatycznie sprawi, że błąd na klasie B **bardziej wpływa na strategię**, więc model będzie się jej uczył z większą uwagą.

## Korzyści:

- Zwiększa **dokładność dla mniejszościowych klas,**
- Poprawia **równowagę predykcji** między wszystkimi emocjami,
- Zapobiega zjawisku **biasu klasy większościowej.**

# PREPROCESSING DANYCH



# PREPROCESSING DANYCH OBRAZOWYCH W PROJEKcie CNN

---

W celu zapewnienia wysokiej jakości uczenia modelu oraz jego dobrej generalizacji na nieznanych danych, zastosowano zaawansowany preprocessing danych obrazowych z wykorzystaniem biblioteki **ImageDataGenerator** z Keras.

## 🎯 Cele preprocessingu:

- Zwiększenie różnorodności danych treningowych (data augmentation),
- Poprawa odporności modelu na zakłócenia (rotacje, przesunięcia, zmiany jasności),
- Zmniejszenie ryzyka **overfittingu** (przeuczenia),
- Dostosowanie obrazów do wymagań architektury sieci.



# PREPROCESSING ZBIORU TRENINGOWEGO

Parametr	Opis transformacji
rescale=1./255	Skaluje wartości pikseli z [0, 255] do [0, 1].
rotation_range=30	Losowa rotacja obrazów o maksymalnie $\pm 30^\circ$ . Pomaga uodpornić model na różne ustawienia kamery.
width_shift_range=0.2	Przesunięcie obrazu w poziomie o maks. 20% szerokości.
height_shift_range=0.2	Przesunięcie obrazu w pionie o maks. 20% wysokości.
zoom_range=0.2	Losowe przybliżenie/oddalenie obrazu do 20%.
shear_range=0.2	Transformacja ścinająca (shear) obrazu. Wprowadza efekt "przechylenia".
horizontal_flip=True	Odbicie poziome obrazu — przydatne np. przy analizie twarzy, emocji itp.
fill_mode='nearest'	Wypełnianie pustych pikseli po transformacjach najbliższymi wartościami sąsiednich pikseli.
brightness_range=[0.8, 1.2]	Jasność obrazu losowo zmieniana w przedziale 80–120%.
channel_shift_range=30.0	Przesunięcie wartości kanałów RGB — symuluje różne oświetlenie lub koloryt skóry.

■ Efekt: Każde wejściowe zdjęcie w treningu może wyglądać inaczej przy każdym przebiegu epoki, co imituje **setki tysięcy wariantów danych!**



# PREPROCESSING ZBIORU WALIDACYJNEGO

Parametr	Opis
rescale=1./255	Obrazy walidacyjne są jedynie skalowane do przedziału [0, 1], bez augmentacji.

## ● Dlaczego bez augmentacji?

Zbiór walidacyjny powinien reprezentować **realne, niezmienione dane**, aby obiektywnie ocenić skuteczność modelu.

# PRÓBA PIERWSZA

---

```
model = Sequential([
    Conv2D(64, (3, 3), activation='relu', input_shape=(48, 48, 1),
kernel_regularizer=regularizers.l2(0.001)),
    BatchNormalization(),
    MaxPooling2D(2, 2),
    Dropout(0.25),

    Conv2D(128, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001)),
    BatchNormalization(),
    MaxPooling2D(2, 2),
    Dropout(0.25),

    Conv2D(256, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001)),
    BatchNormalization(),
    MaxPooling2D(2, 2),
    Dropout(0.25),

    Flatten(),
    Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
    BatchNormalization(),
    Dropout(0.3),

    Dense(7, activation='softmax')
])
```

```
model.compile(optimizer=Adam(learning_rate=0.001),
loss='categorical_crossentropy',
metrics=['accuracy'])

early_stopping =
EarlyStopping(monitor='val_loss', patience=20,
restore_best_weights=True)
```

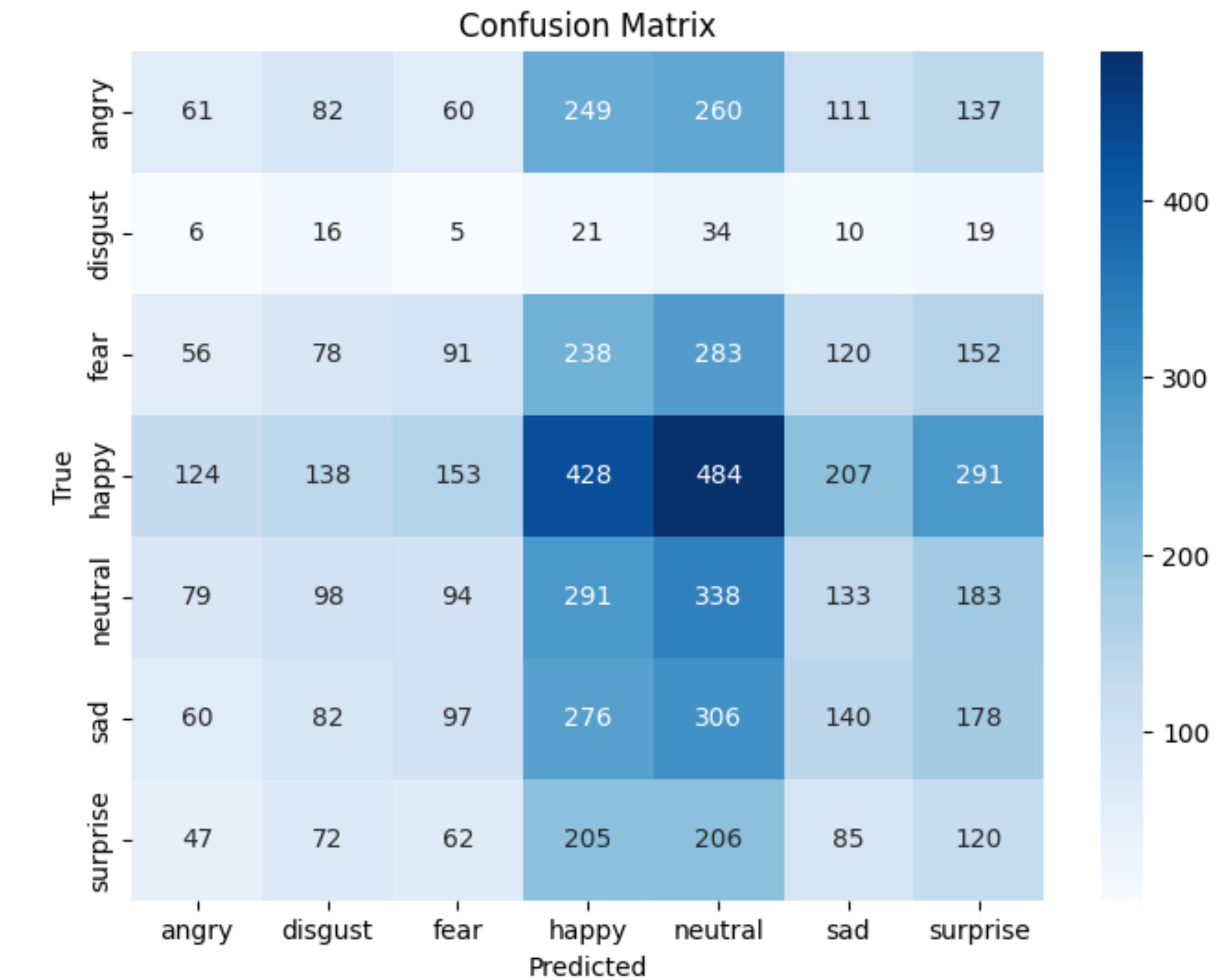
# **KLASYFIKACJA & OPTYMALIZACJA**

# PRÓBA PIERWSZA

LICZBA EPOCH: 100

DŁUGOŚĆ TRENINGU: 10 GODZIN

DOKŁADNOŚĆ (ACCURACY): 0.56425



# PRÓBA DRUGA

---

- Drugi model ma dodatkową warstwę: Conv2D(512, (3, 3), ...).

```
Conv2D(512, (3, 3), activation='relu',
       kernel_regularizer=regularizers.l2(0.001)),
       BatchNormalization(),
       MaxPooling2D(2, 2),
       Dropout(0.3),
```

- Zwiększenie wartości Dropout:

```
Conv2D(512, (3, 3), activation='relu',
       kernel_regularizer=regularizers.l2(0.001)),
       BatchNormalization(),
       MaxPooling2D(2, 2),
       Dropout(0.3), # Zwiększenie dropout
```

```
Flatten(),
Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
BatchNormalization(),
Dropout(0.4), # Zwiększenie dropout
```

- Zmiana współczynnika uczenia:

W drugim modelu learning\_rate optymalizatora Adam zmniejszono do 0.0005 (w porównaniu do 0.001 w pierwszym modelu).

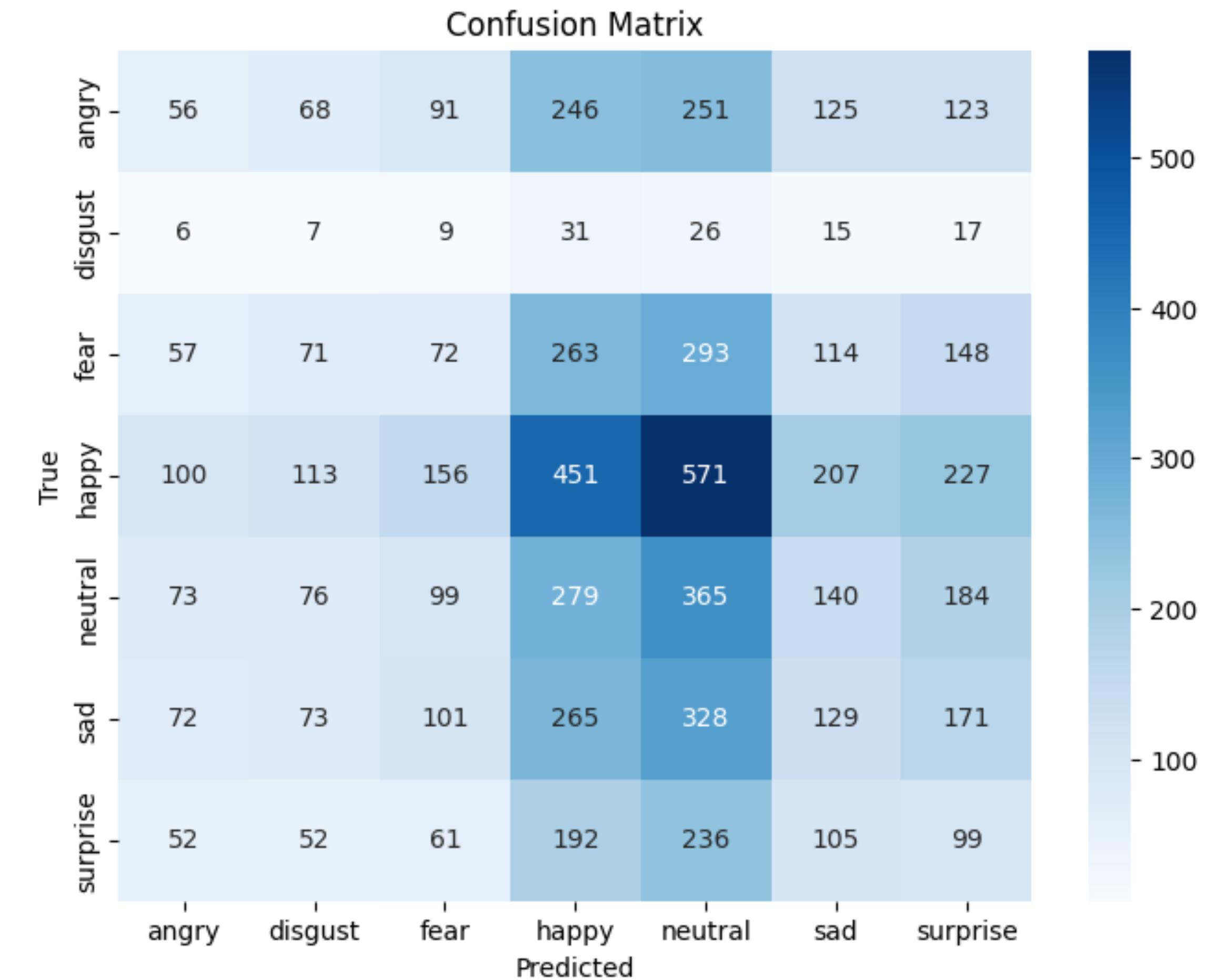
```
model.compile(optimizer=Adam(learning_rate=0.0005),
               loss='categorical_crossentropy', metrics=['accuracy'])
```

# PRÓBA DRUGA

LICZBA EPOCH: 100

DŁUGOŚĆ TRENINGU: 16 GODZIN

DOKŁADNOŚĆ (ACCURACY): 0.59708



# PRÓBA TRZECI

## ► Zmiana optymalizatora:

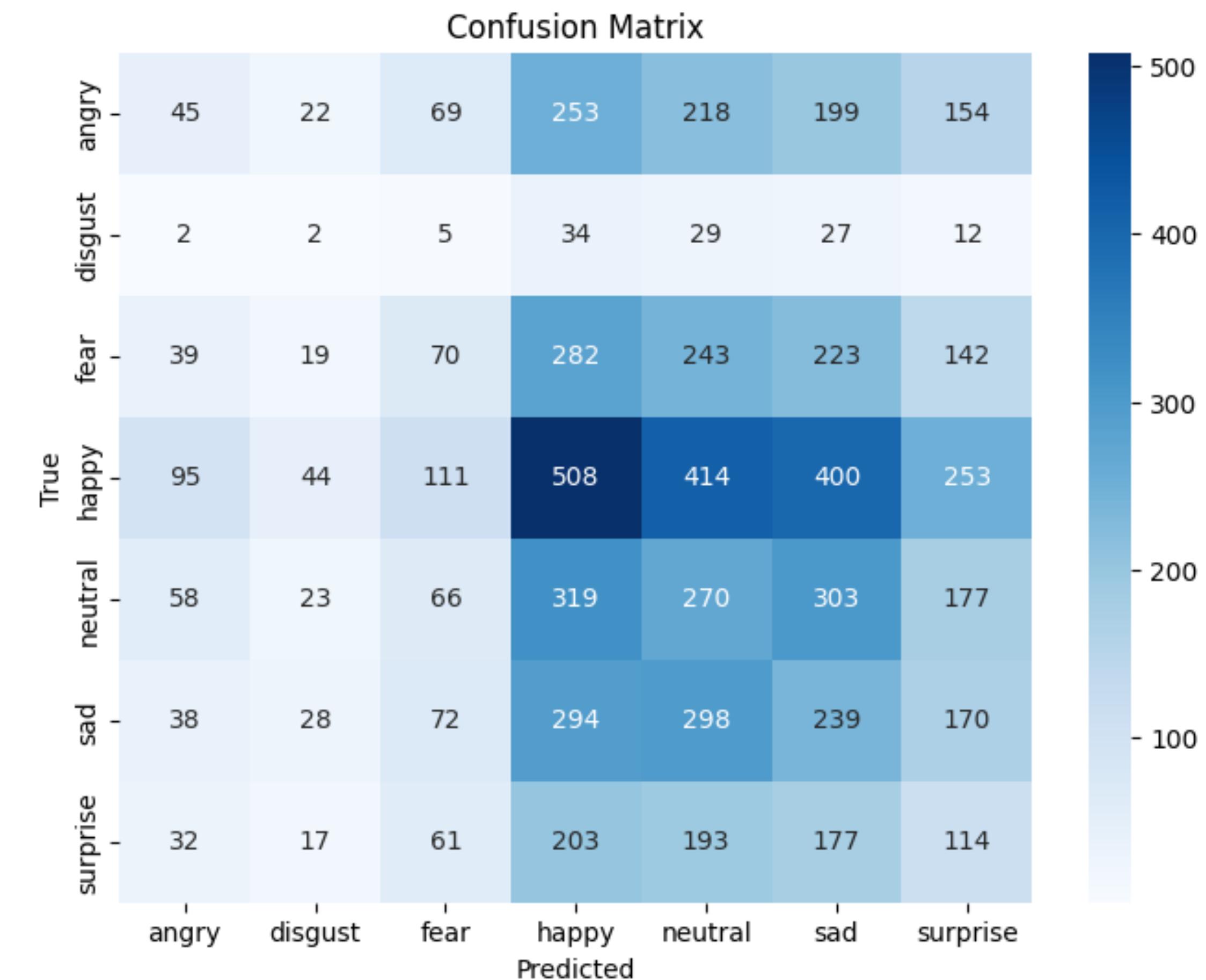
W trzeciej wersji modelu zastąpiono optymalizator **Adam** optymalizatorem **RMSprop**, przy czym współczynnik uczenia pozostał na poziomie 0.001 – takim samym jak w pierwszym modelu.

```
model.compile(optimizer=RMSprop(learning_rate=0.001),  
loss='categorical_crossentropy', metrics=['accuracy'])
```

LICZBA EPOCH: 100

DŁUGOŚĆ TRENINGU: 12 GODZIN

DOKŁADNOŚĆ (ACCURACY): 0.55618



# PRÓBA CZWARTA

## ► Zmiana współczynnika uczenia:

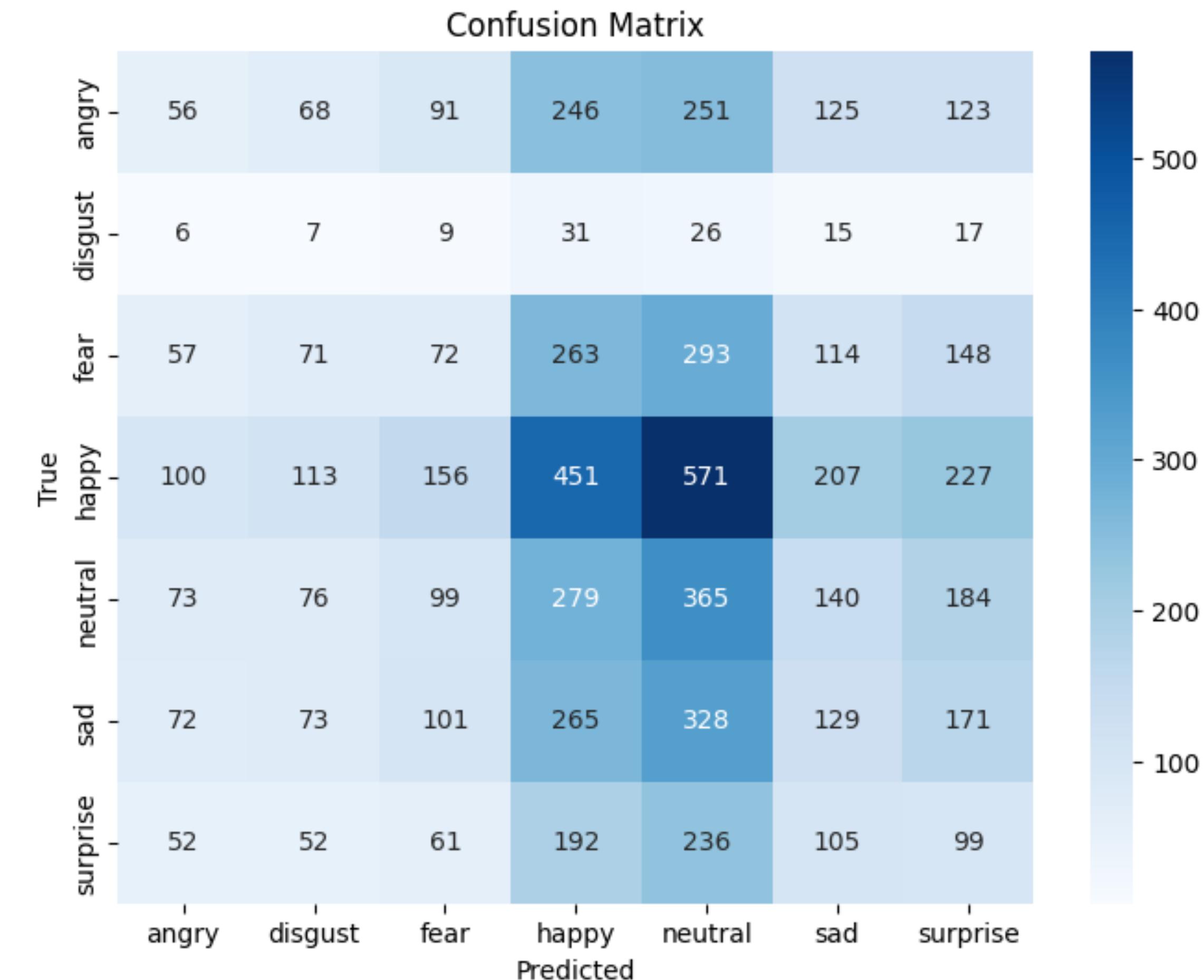
W czwartym modelu learning\_rate optymalizatora **RMSprop** zmniejszono do 0.0005 (w porównaniu do 0.001 w trzecim modelu).

```
model.compile(optimizer=RMSprop(learning_rate=0.0005),  
loss='categorical_crossentropy', metrics=['accuracy'])
```

**LICZBA EPOCH: 100**

**DŁUGOŚĆ TRENINGU: 14 GODZIN**

**DOKŁADNOŚĆ (ACCURACY): 0.52874**



# PRÓBA PIĄTA

---

► **Augmentacja danych:**

1. Dodano: **brightness\_range=[0.8, 1.2]** – losowa zmiana jasności obrazu, co zwiększa odporność modelu na różne warunki oświetleniowe.
2. Dodano: **channel\_shift\_range=30.0** – przesunięcie wartości kanałów kolorystycznych, co uczy model radzenia sobie z różnicami kolorystycznymi.

► **Funkcja aktywacji:**

Dodano warstwę **LeakyReLU(alpha=0.1)** zamiast standardowej funkcji ReLU, co poprawia propagację gradientów i zmniejsza ryzyko występowania „martwych” neuronów – szczególnie w głębszych sieciach konwolucyjnych.

► **Zmiana warstwy spłaszczającej dane:**

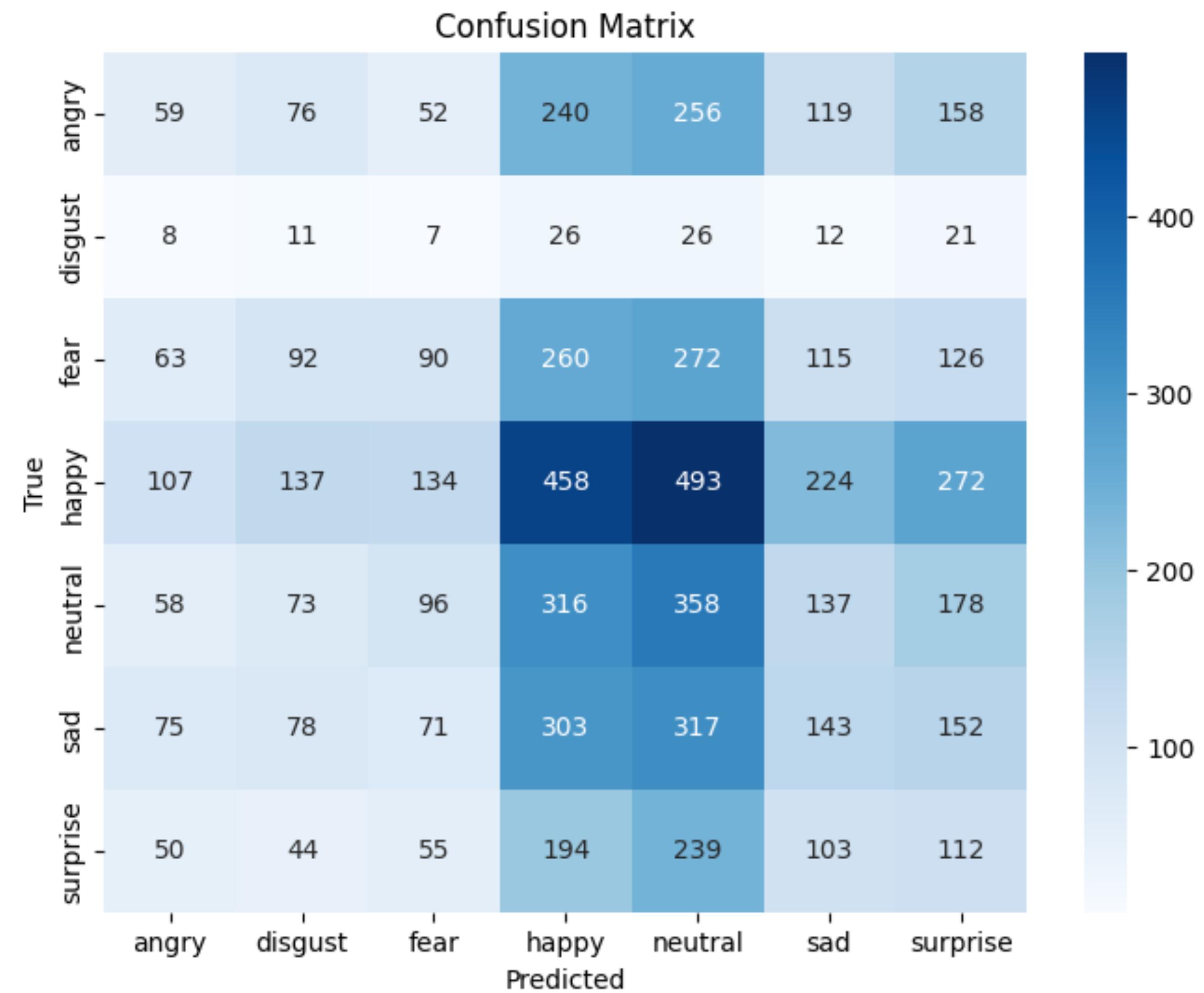
Zamiast Flatten() zastosowano **GlobalAveragePooling2D()** – zmniejsza liczbę parametrów i ogranicza ryzyko przeuczenia, zachowując istotne cechy reprezentacyjne.

# PRÓBA PIĄTA

LICZBA EPOCH: 100

DŁUGOŚĆ TRENINGU: 13 GODZIN

DOKŁADNOŚĆ (ACCURACY): **0.58873**



# PRÓBA SZUSTA

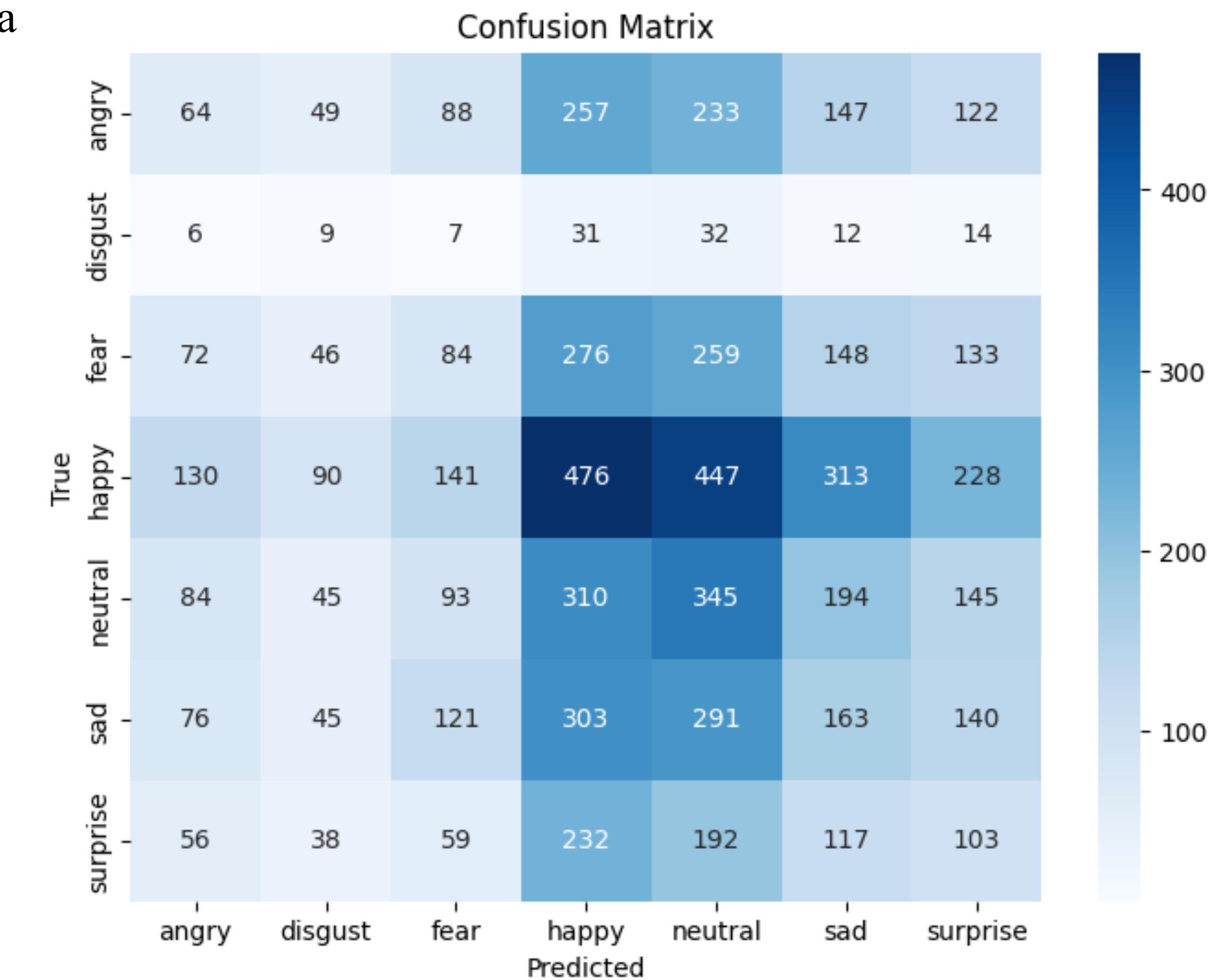
## ► Rozdzielcość wejściowa obrazów:

Zmieniono parametr target\_size na **(96, 96)**, co pozwala modelowi pracować na większych obrazach niż wcześniej (np. 48×48), umożliwiając dokładniejsze wychwytywanie cech i detali mimiki twarzy.

**LICZBA EPOCH: 200**

**DŁUGOŚĆ TRENINGU: 27 GODZIN**

**DOKŁADNOŚĆ (ACCURACY): 0.63119**



# PRÓBA SIÓDMA

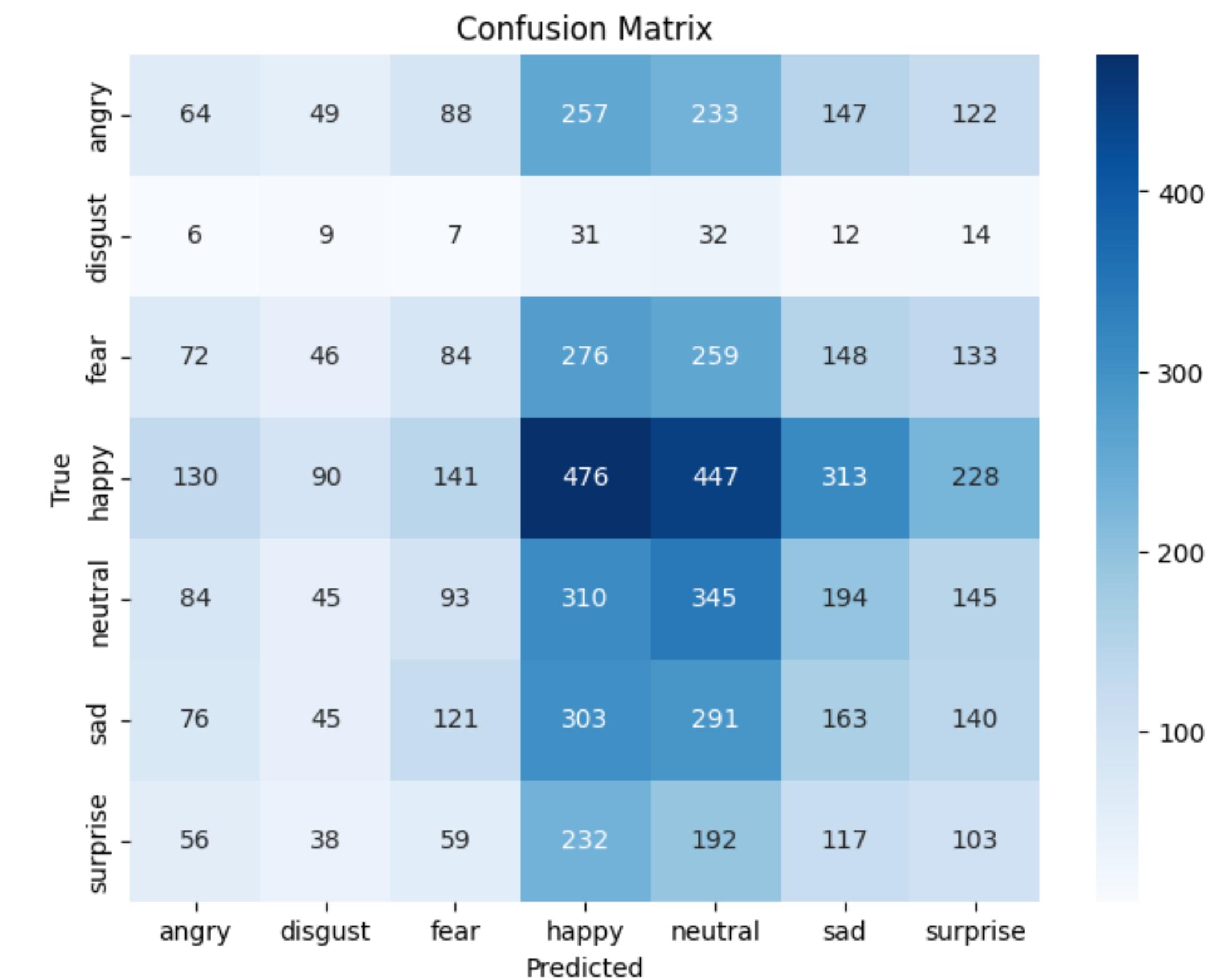
## ► Inicjalizacja wag:

Dodano parametr `kernel_initializer='he_normal'`, który inicjalizuje wagi warstw konwolucyjnych w sposób zoptymalizowany dla aktywacji ReLU. Umożliwia to szybsze i bardziej stabilne uczenie się sieci poprzez lepsze rozłożenie wartości początkowych wag, co może prowadzić do szybszej konwergencji i lepszych wyników modelu.

LICZBA EPOCH: 200

DŁUGOŚĆ TRENINGU: 25 GODZIN

DOKŁADNOŚĆ (ACCURACY): **0.60855**



# PRÓBA ÓSMA

## ➤ Ograniczenie normy gradientu:

Dodano parametr **clipnorm=1.0** do optymalizatora, co ogranicza wartość normy gradientu do 1.0. Pomaga to zapobiec eksplozji gradientów podczas treningu, zwiększając stabilność procesu uczenia, szczególnie w głębszych sieciach lub przy dużych wartościach strat.

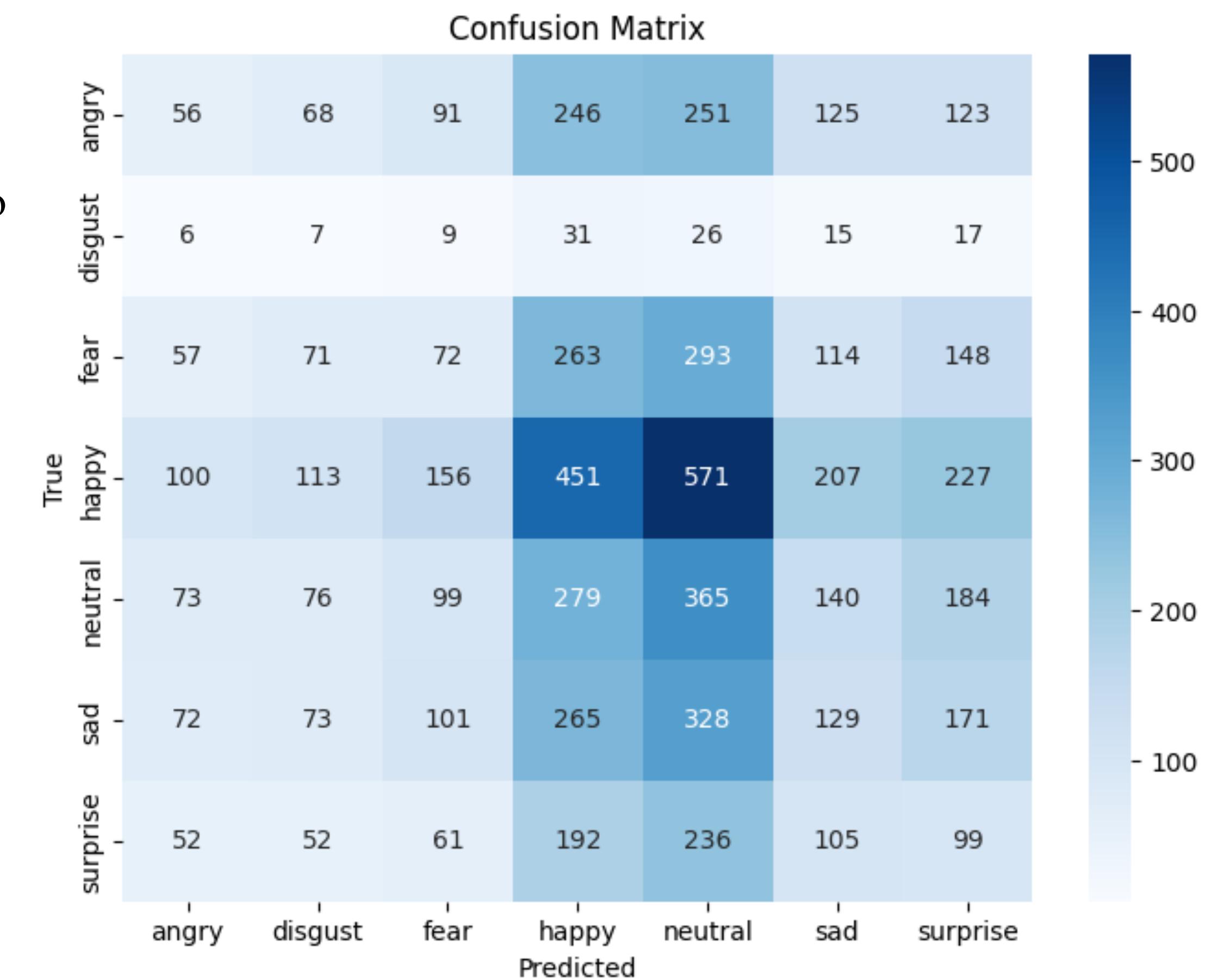
## ➤ Funkcja aktywacji:

Zastosowano funkcję aktywacji **LeakyReLU(alpha=0.1)** zamiast klasycznego ReLU. Pozwala to na przepuszczanie niewielkich ujemnych wartości (z nachyleniem 0.1), dzięki czemu sieć unika problemu „martwych neuronów” i lepiej radzi sobie z propagacją sygnału w głębszych warstwach.

**LICZBA EPOCH: 100**

**DŁUGOŚĆ TRENINGU: 13 GODZIN**

**DOKŁADNOŚĆ (ACCURACY): 0.61336**



# PRÓBA DZIEWIĄTA

---

## ➤ Transfer learning:

Zastosowano transfer learning poprzez wykorzystanie modelu **MobileNetV2** jako warstwy bazowej. MobileNetV2 został załadowany z wstępnie wytrenowanymi wagami na zbiorze ImageNet, co pozwala na wykorzystanie wcześniej wyuczonych reprezentacji cech obrazu. Wagi tej części modelu zostały zamrożone (trainable = False), dzięki czemu trening skupia się wyłącznie na nowo dodanych warstwach klasyfikujących. Rozwiążanie to znaczaco przyspiesza trening i poprawia dokładność na mniejszych zbiorach danych.

## ➤ GlobalAveragePooling2D zamiast Flatten:

Zastosowano warstwę **GlobalAveragePooling2D** w miejsce klasycznego **Flatten**. Warstwa ta redukuje każdy kanał mapy cech do jednej wartości poprzez obliczenie średniej, co zmniejsza liczbę parametrów i ogranicza ryzyko przeuczenia. Ułatwia to także integrację z sieciami transfer learningowymi, które generują duże tensory cech.

## ➤ Preprocessing wejścia:

Zmieniono sposób wczytywania danych z **rescale=1./255** na dedykowaną funkcję **preprocess\_input** dla MobileNetV2. Funkcja ta dostosowuje obrazy do zakresu wartości i statystyk wymaganych przez MobileNetV2, co zapewnia lepszą zgodność danych wejściowych z oczekiwaniemi modelu bazowego.

**LICZBA EPOCH: 100**

**DŁUGOŚĆ TRENINGU: 19  
GODZIN**

**DOKŁADNOŚĆ (ACCURACY):  
0.58631**

# WYNIK

---

 Przeprowadzono serię **8** prób treningu modelu z różnymi konfiguracjami hiperparametrów i architekturą.

 **Kryterium wyboru:**

Najlepszy model wybrano na podstawie **najwyższej dokładności walidacyjnej (val\_accuracy)**.

 **Wybrana próba: Próba nr 6**

- **Dokładność walidacyjna (val\_accuracy):** *np. 63.2%*
- **Dobre uogólnienie na zbiorze walidacyjnym**
- **Brak nadmiernego dopasowania (overfittingu)**

 Model z próby 6 został zapisany do pliku `best_model.h5` dzięki callbackowi `ModelCheckpoint`.

# DOKUMENTACJA APLIKACJI "EMOCJE W AKCJI"

INSPIROWANE FILMEM DISNEY/PIXAR "W GŁOWIE SIĘ NIE MIEŚCI"





# WPROWADZENIE

---

- Uruchom aplikację:** Otwórz aplikację w przeglądarce (wymagany dostęp do kamery i mikrofonu).
- Przygotuj się:** Stań w dobrze oświetlonym miejscu, aby kamera mogła uchwycić Twoją twarz.
- Kliknij "Rozpoznaj emocję":** Aplikacja przeanalizuje wyraz Twojej twarzy za pomocą AI.



# RADOŚĆ (JOY)

**Postać z bajki:**  Joy – energetyczna i pełna pomysłów!

**„Czujesz eksplozję energii? Czas tańczyć!”**

1  Wykonaj zdjęcie przez kamerę aplikacji.

3  Rozpocznij Challenge Tańca:

- Naśladuj ruchy z wideo (skoki, machanie rękami).
- Zdobywaj punkty w ciągu 30 sekund!

4  Muzyka w tle utrzyma Twój rytm.

6  Zobacz wynik i podziel się radością!



# 😢 SMUTEK (SADNESS)

---

**Postać z bajki:** 💡 *Sadness* – wrażliwa i empatyczna.

**„Potrzebujesz wsparcia? Jestem tu dla Ciebie”**

1. 📱 Wykryj smutek przez kamerę.
3. 💬 Porozmawiaj z Botem Wsparcia:
  - Wybierz: ćwiczenia oddechowe 🧘, afirmacje 💬, medytacje 🧘.
3. 🎵 Spokojna muzyka pomoże Ci się zrelaksować.
5. ✨ Zakończ, gdy poczujesz ulgę.





# SMUTEK (SADNESS)

Twoja emocja to



Wykonaj akcję



# 😡 GNIEW (ANGER)

---

**Postać z bajki:** 🔥 *Anger* – impulsywny, ale pełen pasji!

„Czujesz, że zaraz eksplodujesz? Oddychaj!”

1. 📸 Zrób zdjęcie z mocnym marsem na czole.
2. 🎵 Rozpocznij Sesję Oddechową:
  - Podążaj za fazami: **Wdech (4s) → Zatrzymaj (2s) → Wydech (4s).**
3. 🍑 Czerwone cząsteczki unoszące się w tle pomogą Ci skupić się na oddechu.
4. ⏳ Czas trwania: ~3 minut.





# GNIEW (ANGER)

Twoja emocja to



Wykonaj akcję



# 😱 STRACH (FEAR)

---

**Postać z bajki:** 😷 *Fear* – ostrożny, ale opiekuńczy.

**„Czujesz niepokój? Znajdziemy razem bezpieczeństwo!”**

- 1 📸 Wykryj strach przez kamerę.
- 2 🌳 Wejdź w Dialog z Wirtualnym Mentorem:
  - Wybieraj opcje, by oswoić lęk (np. wyobraź sobie spokojny las 🌲).
- 3 ♪ Delikatna muzyka zmniejszy napięcie.
- 5 🔶 Zakończ, gdy poczujesz kontrolę nad emocjami.



# 😱 STRACH (FEAR)

---

Twoja emocja to



Wykonaj akcję

# 😢 ODRAZA (DISGUST)

---

**Postać z bajki:** 🌱 *Disgust* – wybredna, ale dbająca o harmonię.

**„Coś Cię zniesmaczyło? Zbadaj swoją reakcję!”**

1. 📸 **Zrób zdjęcie** z grymasem niesmaku.

2. 🖼 **Rozwiąż Test Rorschacha:**

- Opisz, co widzisz na abstrakcyjnych plamach (np. „owady” 🕿 lub „kwiaty” 🌸).

3. 📈 **Otrzymaj Podsumowanie:**

- Np. „Widzisz pustkę? To naturalna obrona umysłu!“.

4. 🎵 **Hipnotyzująca muzyka** towarzyszy testowi.





## ODRAZA (DISGUST)

---

Twoja emocja to



Wykonaj akcję

# 😐 NEUTRALNOŚĆ I 😲 ZASKOCZENIE (NIEOBECNE W FILMIE!)

---

1. 📸 Wykryj neutralność lub zaskoczenie przez kamerę aplikacji.

## 3. II Brak dedykowanej aktywności:

- Emocja neutralna nie ma odpowiadającej interakcji w aplikacji (nie występowała w filmie 🎬).

## 3. ⭐ Co dalej?:

- Ciesz się chwilą spokoju lub wybierz inną emocję!



😐 NEUTRALNOŚĆ I 😲 ZASKOCZENIE (NIEOBECNE W FILMIE!)

---

Twoja emocja to

neutral



Wykonaj akcję

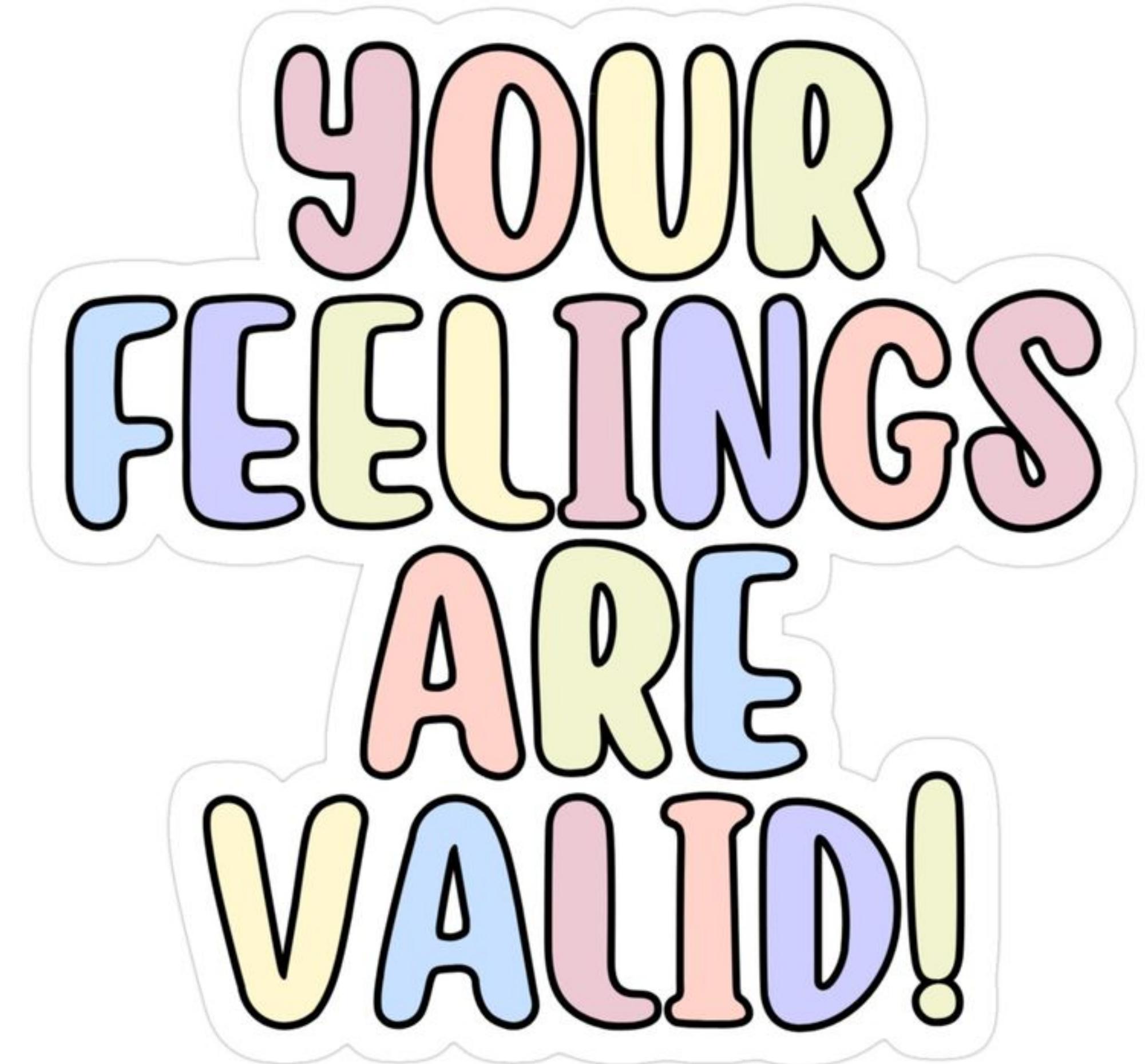


# KAŻDA EMOCJA JEST WAŻNA!

„Jak w filmu – emocje tworzą Ciebie! ❤”

- ⌚ **Eksperymentuj:** Wracaj do aplikacji, gdy zmieniasz nastrój.
- 🎭 **Wyrażaj się:** Taniec, oddech, rozmowa – wybierz, co działa dla Ciebie.
- 🎵 **Muzyka:** Każda emocja ma własną ścieżkę dźwiękową!

„Dziękuję, że jesteś sobą! 🧠✨”



„Dziękuję, że jesteś sobą! 🧠✨”