```
y = y[mask]

# --- [4] Tambahkan channel dimensi untuk CNN ---
X = X[..., np.newaxis]

# --- [5] Bagi data train/test ---
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# --- [6] Load model CNN ---
model = load_model("model33.h5")

# --- [7] Prediksi ---
y_pred_proba = model.predict(X_test)
y_pred = np.argmax(y_pred_proba, axis=1)

# --- [8] Confusion Matrix ---
labels = ["Horizontal", "Imbalance", "Normal", "Vertical"]
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot(cmap="Blues")
plt.title("Confusion Matrix CNN - MFCC 33")
plt.show()

# --- [9] Classification Report ---
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=labels))

# --- [10] ROC Curve dan AUC ---
n_classes = len(kelas_diambil)
y_test_bin = label_binarize(y_test, classes=kelas_diambil)
```

```
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_pred_proba[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

plt.figure()
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f"{labels[i]} (AUC = {roc_auc[i]:.2f})")

plt.plot([0, 1], [0, 1], "k--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve CNN - MFCC 33")
plt.legend(loc="lower right")
plt.grid(True)
plt.show()

# --- [11] AUC Summary ---
print("\nAUC Score per Class:")
for i in range(n_classes):
    print(f"{labels[i]}: AUC = {roc_auc[i]:.4f}")

macro_auc = roc_auc_score(y_test_bin, y_pred_proba[:, :n_classes],
average="macro")
weighted_auc = roc_auc_score(y_test_bin, y_pred_proba[:, :n_classes],
average="weighted")
```