German Traffic Sign Classification Using TensorFlow

ARPAN PRADHAN

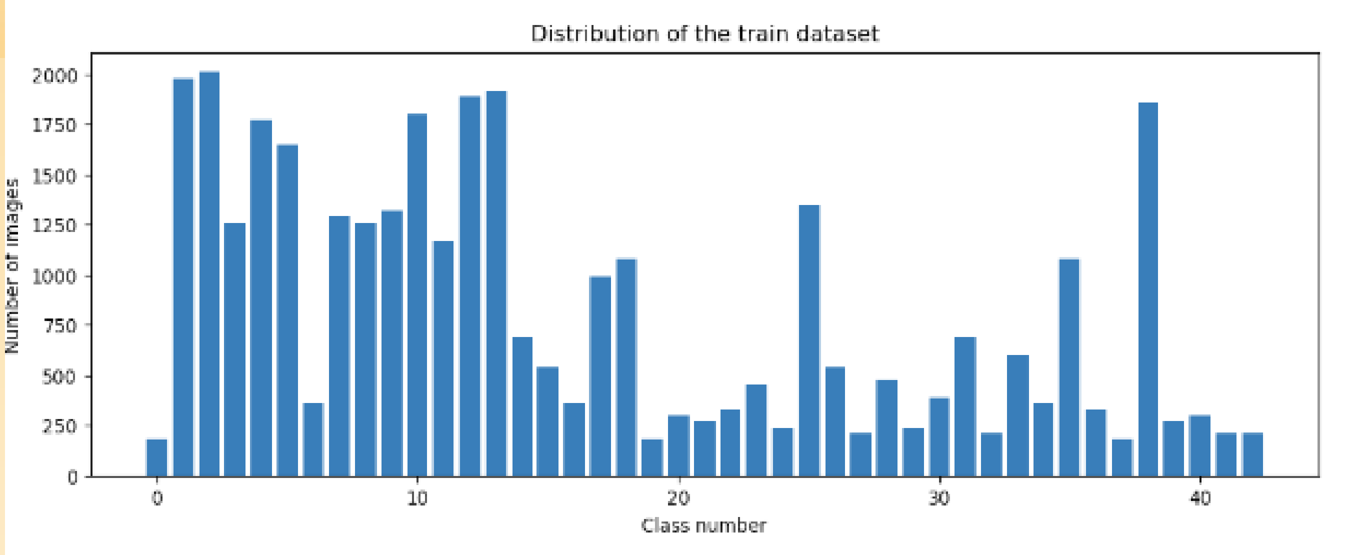
5-18-2024

Problem Statement :

- THE GERMAN TRAFFIC SIGN RECOGNITION BENCHMARK (GTSRB) DATASET PRESENTS A CHALLENGE IN ACCURATELY CLASSIFYING TRAFFIC SIGN IMAGES, WHICH IS CRUCIAL FOR DEVELOPING ROBUST AUTONOMOUS DRIVING SYSTEMS. WITH OVER 50,000 LABELED IMAGES ACROSS 43 CLASSES, THE DATASET POSES A CLASSIFICATION TASK WITH SIGNIFICANT COMPLEXITY DUE TO VARIATIONS IN LIGHTING CONDITIONS, OCCLUSIONS, AND GEOMETRIC TRANSFORMATIONS.

- TO ADDRESS THIS CHALLENGE, THE TASK INVOLVES IMPLEMENTING THE LENET CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE TO CLASSIFY TRAFFIC SIGN IMAGES WITH HIGH ACCURACY AND EFFICIENCY.

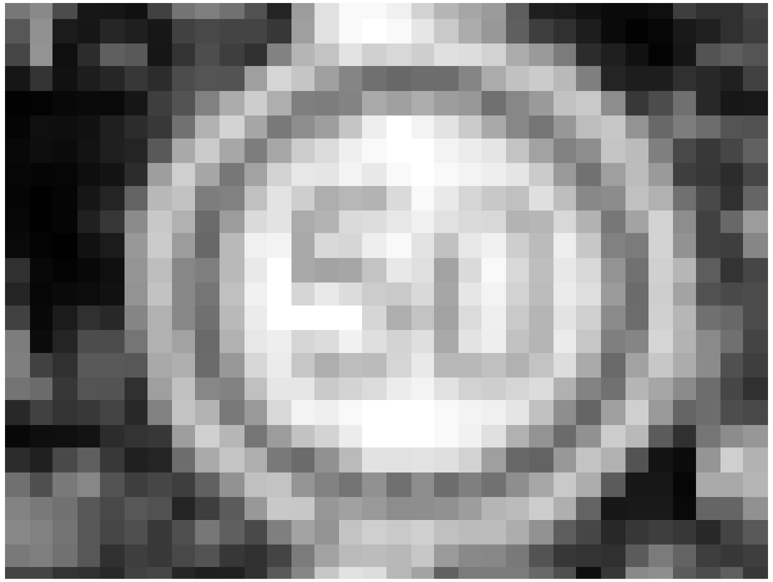
- LIMITED DATA PER CLASS: DATASET CONTAINS 43 CLASSES, UNUNIFORMLY DISTRIBUTED WITHIN EACH CLASS.
- CLASS IMBALANCE: SOME CLASSES HAD SIGNIFICANTLY FEWER SAMPLES, CAUSING IMBALANCED CLASS DISTRIBUTIONS.
- MODEL OVERFITTING: RISK OF OVERFITTING DUE TO SMALLER DATASET SIZE, ESPECIALLY FOR CLASSES WITH FEWER INSTANCES.
- FEATURE REPRESENTATIONS: DESPITE LIMITED DATA, THE



Non-Uniform distribution of train dataset

Train images

- GRAYSCALE CONVERSION: CONVERTED RGB IMAGES TO GRAYSCALE
- HISTOGRAM EQUALIZATION: ENHANCED IMAGE CONTRAST AND STANDARDIZE LIGHTING.
- EXAMPLE CODE: IMG = CV2.EQUALIZEHIST(IMG)
- ENHANCED IMAGE CONTRAST AND STANDARDIZE LIGHTING.
- EXAMPLE CODE: IMG = CV2.EQUALIZEHIST(IMG)
- ONE-HOT ENCODING: CONVERTED CATEGORICAL LABELS TO BINARY VECTORS.
- NORMALIZATION: SCALE PIXEL VALUES TO [0, 1] RANGE FOR MODEL TRAINING.
- EXAMPLE CODE: IMG = IMG / 255
- DATA SHAPE : RESHAPED DATA TO THE REQUIRED FORMAT FOR MODEL TRAINING.



Baseline Model

```
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.optimizers import Adam
def leNet_model():
    model = Sequential()
    model.add(Conv2D(60, (5, 5), input_shape=(32, 32, 1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(15, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(500, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(Adam(learning_rate=0.01), loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

```
model = leNet_model()
print(model.summary())
```

Model: "sequential_5"

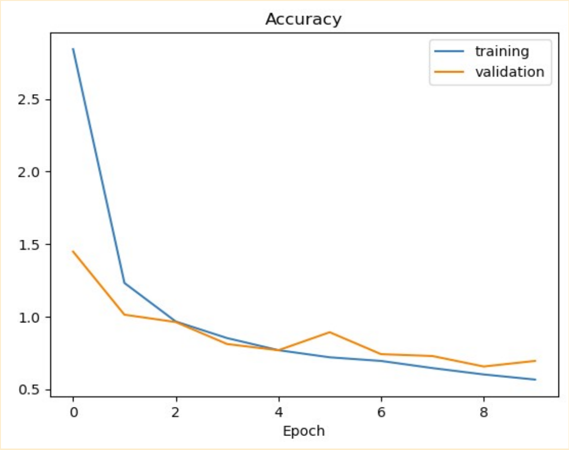
Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 28, 28, 60)	1,560
max_pooling2d_10 (MaxPooling2D)	(None, 14, 14, 60)	0
conv2d_11 (Conv2D)	(None, 12, 12, 15)	8,115
max_pooling2d_11 (MaxPooling2D)	(None, 6, 6, 15)	0
flatten_5 (Flatten)	(None, 540)	0
dense_10 (Dense)	(None, 500)	270,500
dropout_5 (Dropout)	(None, 500)	0
dense_11 (Dense)	(None, 43)	21,543

Total params: 301,718 (1.15 MB)

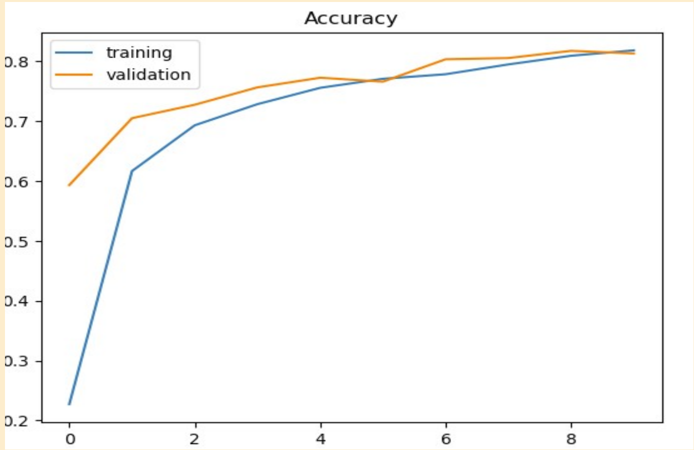
Trainable params: 301,718 (1.15 MB)

Non-trainable params: 0 (0.00 B)

None

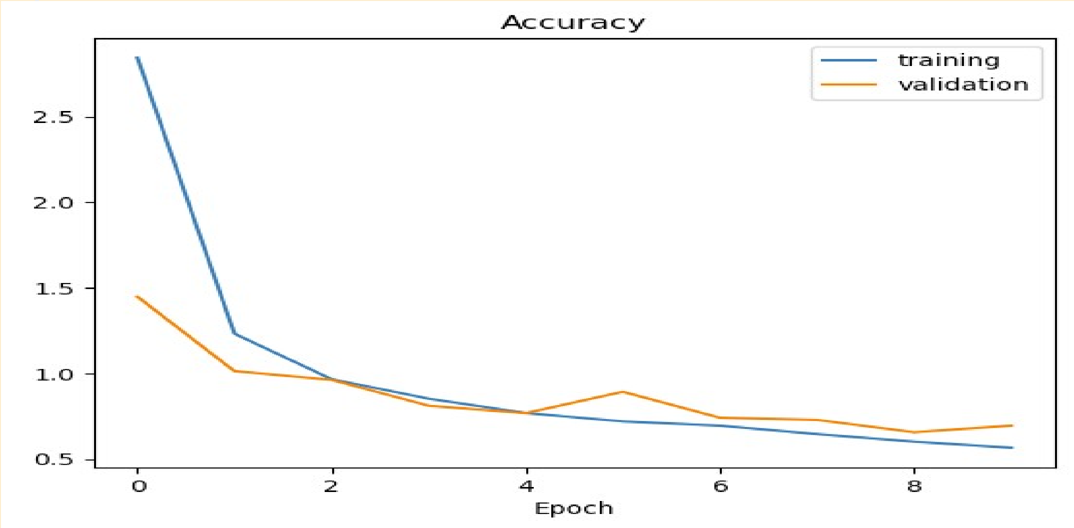


Training vs Validation Loss

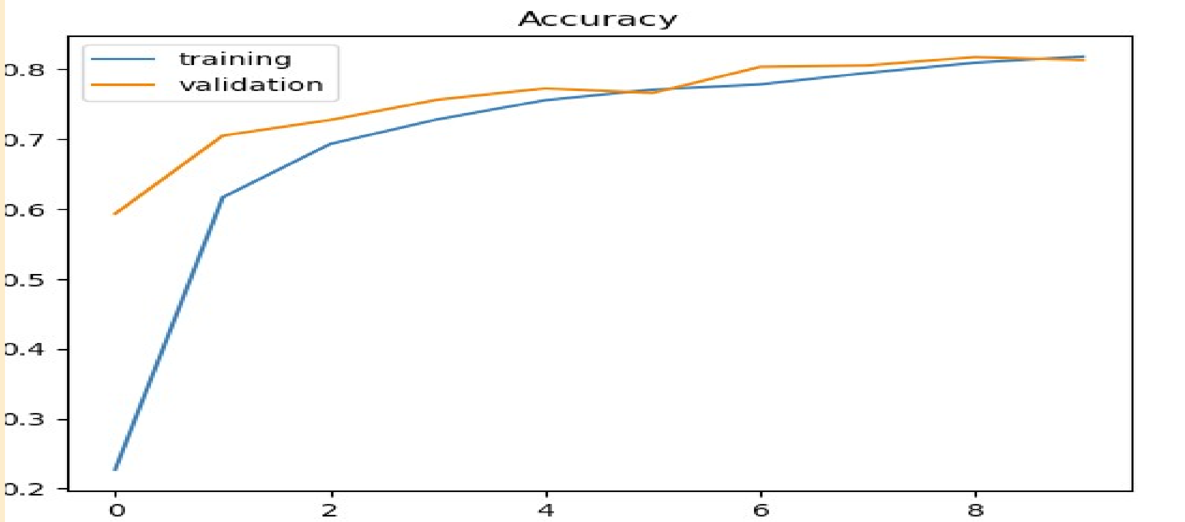


Training vs Validation Accuracy

- OVERFITTING: SIGNIFICANT DISCREPANCY BETWEEN TRAINING AND VALIDATION LOSSES
- TEST ACCURACY: ACHIEVED 80%
- VALIDATION ACCURACY: COMPARABLE TO TEST ACCURACY
- UNDERFITTING: ELEVATED TRAINING AND VALIDATION LOSS



Training vs Validation Loss

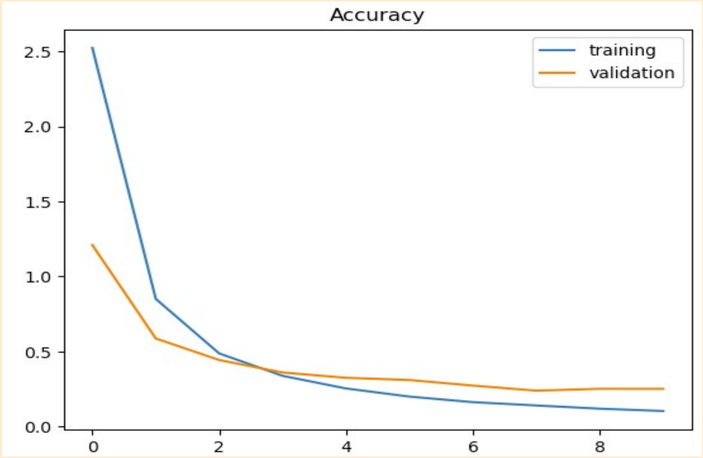


Training vs Validation Loss

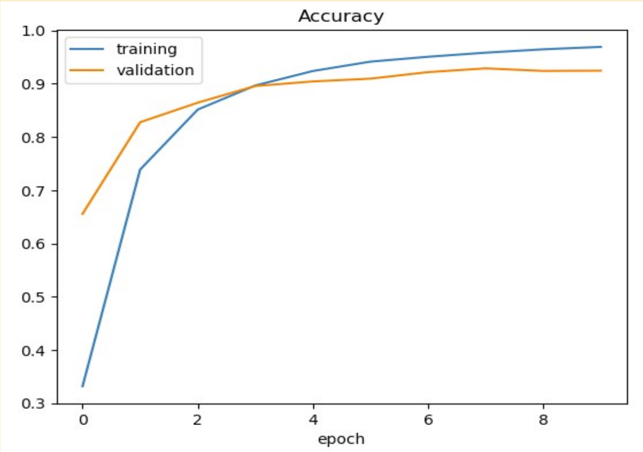
Metric	Train	Validation	Testing
Loss	0.34	0.69	0.76
Accuracy	0.89	0.81	0.80

Impact of Learning Rate on Model Accuracy

- ORIGINAL LEARNING RATE: 0.01
- MODEL PERFORMANCE: TEST ACCURACY OF 80%, TEST LOSS OF 0.762
- HIGHER LEARNING RATE LED TO OVERSHOOTING OR INSTABILITY DURING TRAINING
- RESULTS: LOWER LEARNING RATE FACILITATED SMOOTHER CONVERGENCE AND BETTER GENERALIZATION.
- ADJUSTING LEARNING RATE FROM 0.01 TO 0.001 RESULTED IN A NOTABLE IMPROVEMENT IN MODEL ACCURACY.



Training vs Validation Loss



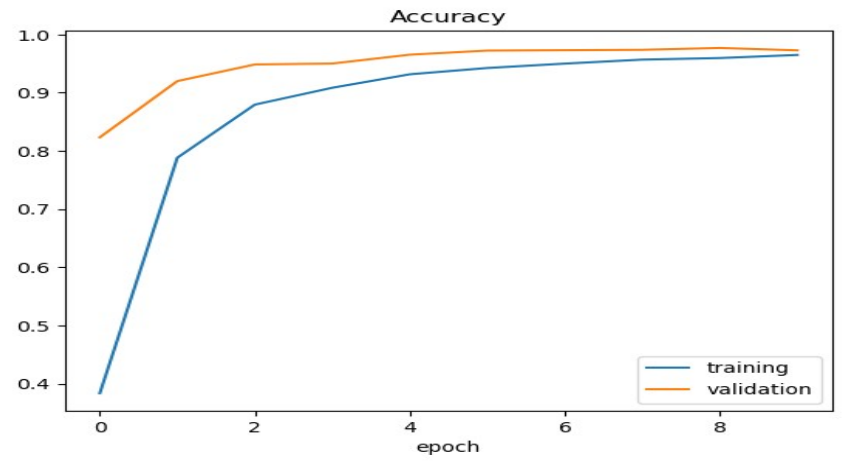
Training vs Validation Accuracy

Metric	Train	Validation	Testing
Loss	0.04	0.25	0.35
Accuracy	0.99	0.92	0.90

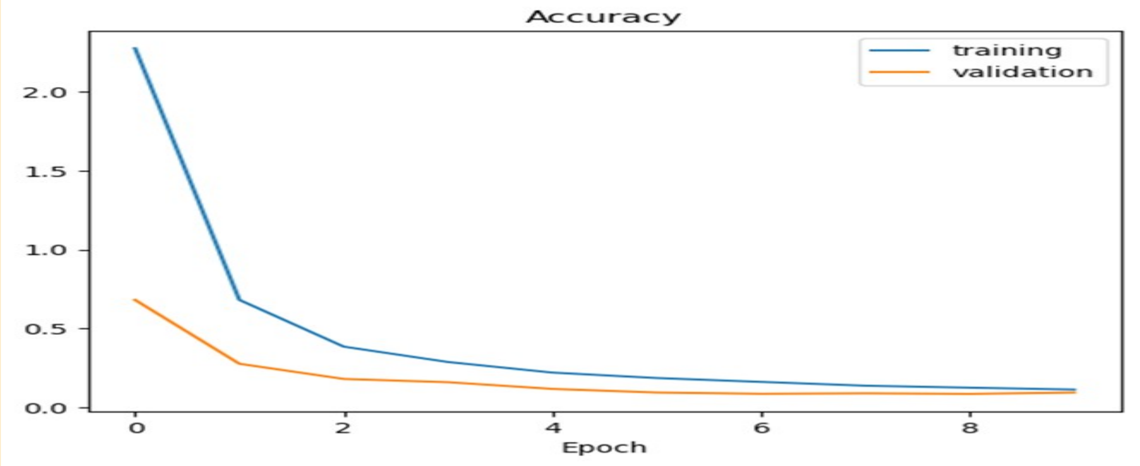
Model Enhancement with Convolutional Layers and Dropout

- ADDED 2 CONVOLUTIONAL LAYERS TO ALLOW FEATURE EXTRACTION
- 2 DROPOUT LAYERS INTRODUCED AFTER EACH CONVOLUTIONAL LAYER TO PREVENT OVERFITTING.
- PROGRESSION OF LOSS AND ACCURACY METRICS THROUGHOUT TRAINING EPOCHS.

Metric	Train	Validation	Test
Loss	0.20	0.09	0.14
Accuracy	0.99	0.97	0.95



Training vs Validation Loss



Training vs Validation Accuracy

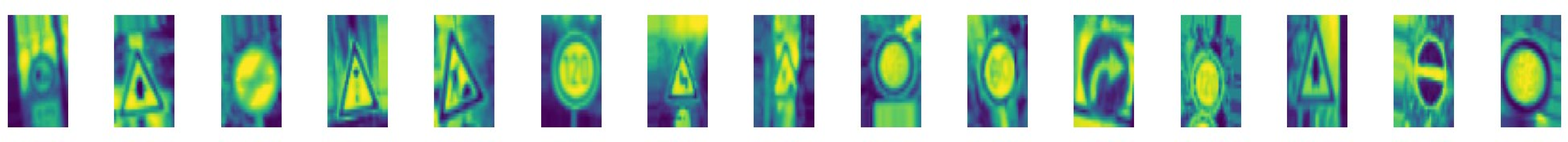
Model Prediction Analysis

- MODEL INACCURATELY PREDICTED "PRIORITY ROAD" FOR THE ACTUAL CLASS "SPEED LIMIT 30KPH".
- MISCLASSIFICATION HIGHLIGHTS A POTENTIAL LIMITATION IN THE MODEL'S ABILITY TO DISTINGUISH BETWEEN SIMILAR TRAFFIC SIGN CLASSES
- PROPOSED SOLUTION : AUGMENT TRAINING DATA WITH ADDITIONAL EXAMPLES OF SIMILAR SIGN CLASSES TO IMPROVE MODEL DIFFERENTIATION.

- AUGMENTATION BOOSTS DATASET SIZE AND EXPOSES MODELS TO VARIED PERSPECTIVES, AIDING ROBUST FEATURE EXTRACTION.

Data Augmentation

- KERAS ‘IMAGEDATAGENERATOR’ TO DEFINE AND APPLY TRANSFORMATIONS LIKE ROTATION, ZOOM, SHEAR
- INCORPORATE AUGMENTED DATA INTO MODEL TRAINING USING ‘MODEL.FIT_GENERATOR’
- MODEL ACHIEVED OVER 99% TRAINING ACCURACY, INDICATING IMPROVED LEARNING CAPABILITY.
- TEST AND VALIDATION ACCURACY ALSO INCREASES, SHOWCASING ENHANCED GENERALIZATION.

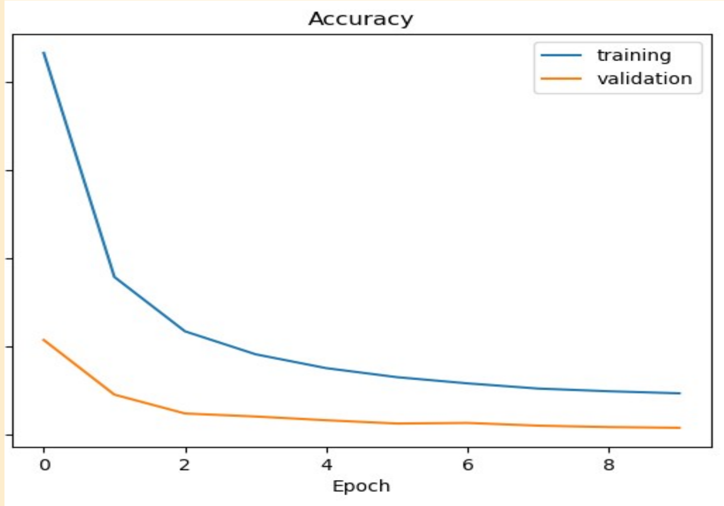


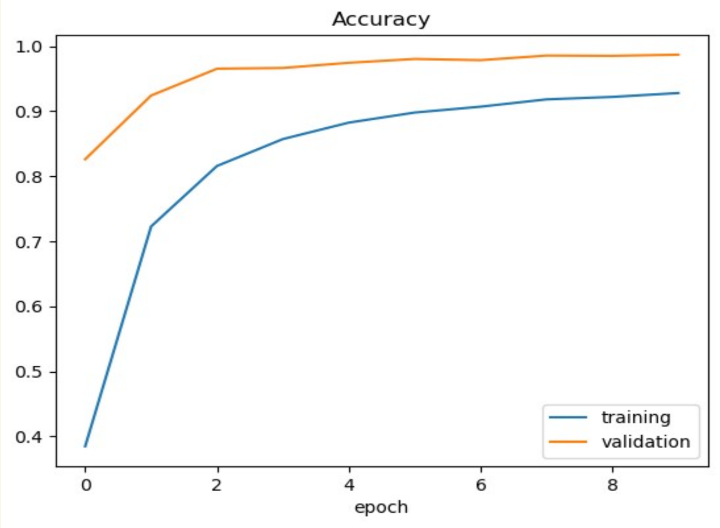
Augmented Images

Model Performance After Data Augmentation

- NOTABLE DISCREPANCY BETWEEN TRAINING LOSS (0.024) AND VALIDATION LOSS (0.040), INDICATING POTENTIAL OVERFITTING TO THE TRAINING DATA.
- SIGNIFICANT INCREASE IN TEST LOSS (0.091) COMPARED TO BOTH TRAINING AND VALIDATION LOSSES, SUGGESTING THE MODEL'S GENERALIZATION ABILITY IS COMPROMISED.
- THE HIGH TRAINING (0.99) AND VALIDATION ACCURACY (0.98) DEMONSTRATE THE MODEL'S EXCELLENT FIT TO THE TRAINING DATA AND REASONABLE FIT TO THE VALIDATION DATA.
- THE VALIDATION ACCURACY (0.98) BEING CLOSE TO THE TEST ACCURACY (0.96) SUGGESTS CONSISTENT MODEL PERFORMANCE ACROSS UNSEEN DATA

Metric	Train	Validation	Test
Loss	0.024	0.040	0.091
Accuracy	0.99	0.98	0.96





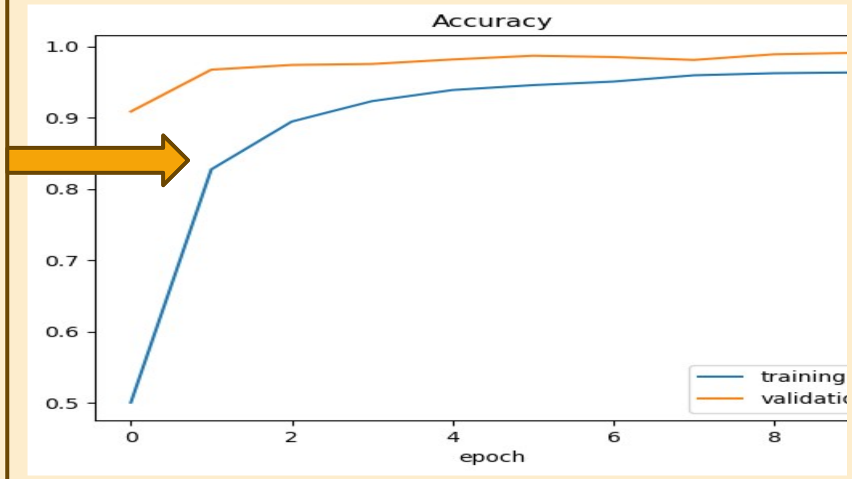
PROPOSED SOLUTION

- DROPOUT LAYERS MAY OVERLY REGULARIZE THE MODEL, LEADING TO UNDERFITTING.
- REMOVE ONE DROPOUT LAYER
- IMPROVED CONVERGENCE AND MODEL PERFORMANCE MATRICES

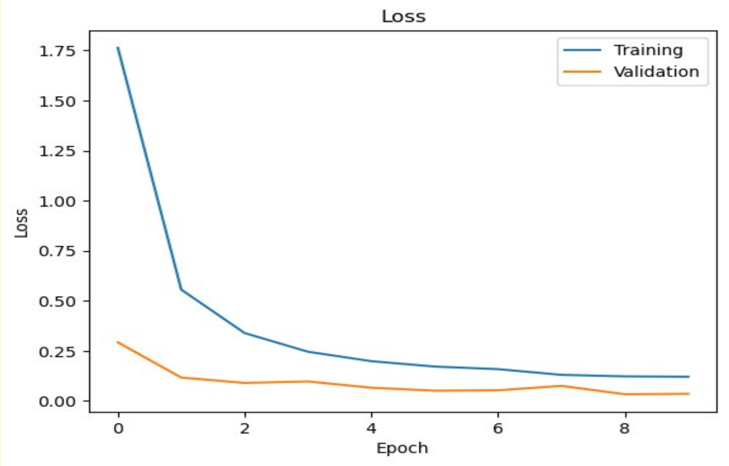
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam

def modified_model():
    model = Sequential()
    model.add(Conv2D(60, (5, 5), input_shape=(32, 32, 1), activation='relu'))
    model.add(Conv2D(60, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(30, (3, 3), activation='relu'))
    model.add(Conv2D(30, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(500, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(43, activation='softmax'))

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

model = modified_model()
print(model.summary())
```

Final Result

- MODEL FIT:
- THE SMALL GAP BETWEEN TRAINING AND VALIDATION LOSS SUGGESTS THE MODEL HAS ACHIEVED A GOOD FIT WITHOUT OVERFITTING, DEMONSTRATING ROBUST LEARNING AND GENERALIZATION CAPABILITIES.
- THE CLOSE ALIGNMENT OF VALIDATION AND TEST LOSSES, AND HIGH ACCURACY METRICS ACROSS ALL DATASETS, INDICATE THAT THE MODEL IS NOT ONLY WELL-TUNED BUT ALSO CONSISTENT AND RELIABLE WHEN APPLIED TO NEW DATA
- THE APPLICATION OF VARIOUS TECHNIQUES HAS EFFECTIVELY MINIMIZED OVERFITTING AND UNDERFITTING, RESULTING IN A BALANCED AND GENERALIZABLE MODEL.



Training vs Validation Accuracy



Training vs Validation loss

Metric	Train	Validation	Test
Loss	0.01	0.03	0.035
Accuracy	0.99	0.99	0.97

Web Image Prediction Results.

```python
import numpy as np
import cv2

# URL of the image
url = 'https://c8.alamy.com/comp/J2MRAJ/german-road-sign-bicycles-crossing-J2MRAJ.jpg'

# Download and load the image
r = requests.get(url, stream=True)
img = Image.open(r.raw)
plt.imshow(img, cmap=plt.get_cmap('gray'))

# Convert the image to grayscale, resize, and preprocess
img = np.asarray(img)
img = cv2.resize(img, (32, 32))
img = preprocess(img)
img = img.reshape(1, 32, 32, 1)

# Predict the class probabilities
predictions = model.predict(img)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
Predicted class: 29
```

- APPLIED THE TRAINED MODEL TO PREDICT THE CLASS LABELS OF WEB IMAGES.
- EVALUATED MODEL PERFORMANCE ON FIVE WEB IMAGES.

Results (Accuracy) and Learnings from the methodology

- ACHIEVED 97.52% TEST ACCURACY THROUGH THE ADOPTION OF A DEEP LEARNING FRAMEWORK, DESPITE THE INTRICATE NATURE OF THE DATASET.
- ADVANCED DATA AUGMENTATION TECHNIQUES, SUCH AS ROTATION, ZOOMING, AND SHEARING, WERE INSTRUMENTAL IN ENRICHING THE DATASET, THEREBY ENHANCING THE MODEL'S GENERALIZATION CAPABILITIES.
- STRATEGIES TO MITIGATE OVERFITTING AND UNDERFITTING WERE RIGOROUSLY IMPLEMENTED, INCLUDING THE INCORPORATION OF CONVOLUTIONAL LAYERS AND FINE-TUNING OF LEARNING RATES TO OPTIMIZE GRADIENT DESCENT.
- DILIGENT REMOVAL OF DROPOUT LAYERS FURTHER REFINED THE MODEL'S ARCHITECTURE, CONTRIBUTING TO IMPROVED PERFORMANCE METRICS.
- THESE COMBINED EFFORTS RESULTED IN A SUBSTANTIAL BOOST IN THE MODEL'S ROBUSTNESS AND ADAPTABILITY, AFFIRMING THE EFFICACY OF THE DEEP LEARNING APPROACH IN TACKLING COMPLEX DATASETS

Future work

- ENHANCED DATA COLLECTION: EMPLOY ACTIVE LEARNING STRATEGIES TO ITERATIVELY COLLECT NEW SAMPLES WHICH FALLS WITHIN UNDERREPRESENTED CLASSES. UTILIZE TECHNIQUES LIKE CROWD-SOURCING OR DOMAIN ADAPTATION TO GATHER DIVERSE DATA ACROSS DIFFERENT GEOGRAPHIC REGIONS AND ENVIRONMENTAL CONDITIONS, ENSURING COMPREHENSIVE COVERAGE OF REAL-WORLD SCENARIOS.
- FINE-TUNING ARCHITECTURES: CONDUCT SYSTEMATIC HYPERPARAMETER TUNING EXPERIMENTS USING RANDOM SEARCH OR BAYESIAN OPTIMIZATION TO OPTIMIZE NETWORK ARCHITECTURES.
- TRANSFER LEARNING: INVESTIGATE DOMAIN ADAPTATION METHODS SUCH AS DOMAIN ADVERSARIAL TRAINING OR GRADIENT REVERSAL LAYERS TO ADAPT PRE-TRAINED MODELS FROM RELATED DOMAINS (E.G., GENERAL OBJECT RECOGNITION) TO THE SPECIFIC TASK OF TRAFFIC SIGN CLASSIFICATION.
- SEMANTIC SEGMENTATION: IMPLEMENT STATE-OF-THE-ART SEMANTIC SEGMENTATION ARCHITECTURES LIKE U-NET OR DEEPLABV3+ TAILORED TO TRAFFIC SIGN SEGMENTATION TASKS.
- REAL-TIME DEPLOYMENT: OPTIMIZE MODEL ARCHITECTURES FOR DEPLOYMENT ON RESOURCE-CONSTRAINED PLATFORMS BY EMPLOYING TECHNIQUES LIKE MODEL QUANTIZATION, WEIGHT PRUNING, OR KNOWLEDGE DISTILLATION.
- DEPLOYMENT IN AUTONOMOUS VEHICLES: DEVELOP CUSTOMIZED INFERENCE PIPELINES TAILORED TO THE LATENCY AND THROUGHPUT REQUIREMENTS OF AUTONOMOUS DRIVING SYSTEMS, INCORPORATING TECHNIQUES LIKE MODEL PIPELINING OR ASYNCHRONOUS PROCESSING TO MAXIMIZE UTILIZATION OF COMPUTATIONAL RESOURCES.

REFERENCES:

SLIM, J. (N.D.). GERMAN TRAFFIC SIGNS. BITBUCKET. RETRIEVED FROM HTTPS://BITBUCKET.ORG/JADSLIM/GERMAN-TRAFFIC-SIGNS

GITHUB LINK: HTTPS://GITHUB.COM/ARPANUCHICAGO/GERMAN-TRAFFIC-SIGN-CLASSIFICATION