YİĞİT SARIOĞLU
2022400354

ARTIFICIAL INTELLIGENCE IN HEALTHCARE

Assignment 4 : Improving the Results

Introduction

In this assignment, my focus revolves around enhancing the performance of the model through the exploration and implementation of various techniques. Through this iterative process of analysis and improvement, my objective is to elevate the accuracy of our AI model, fostering a deeper understanding of the intricate interplay between data, model architecture, and optimization strategies. In this report, first part of it, I gave general information about the reason for low accuracy. In the last part, I commented on my experiments and their results.

1. Possible Reasons For Low Accuracy and Its Solutions

a. Possible Reasons for Low Accuracy:

There are many potential causes of low accuracy when training a machine learning model. Some common causes include:

1. Insufficient or poor-quality data: One of the most common causes of low accuracy is having insufficient or poor-quality data to train the model. This can happen if the dataset is too small, or if it is biased or contaminated in some way. In these cases, the model may not have enough information to learn from, or it may be learning from misleading or irrelevant data, which can lead to low accuracy.
2. Overfitting: Overfitting is another common cause of low accuracy. Overfitting occurs when a model is too complex and learns too much from the training data, causing it to perform poorly on unseen data. This can be caused by having too many parameters in the model, or by using a model that is not well suited to the problem.
3. Underfitting: Underfitting is the opposite of overfitting, and occurs when a model is too simple and does not learn enough from the training data. This can also lead to low accuracy, as the model may not be able to capture the complexity and variability of the data.
4. Poor model selection: Choosing the wrong model for the problem can also lead to low accuracy. Different models have different strengths and

weaknesses, and it is important to choose a model that is well suited to the problem at hand.

5. Lack of feature engineering: Feature engineering is the process of selecting and designing features for a machine learning model. If the features used to train the model are not well designed or relevant, the model may have difficulty learning from the data and may have low accuracy.
6. Hyperparameter tuning: Hyperparameters are the parameters that control the behavior of a machine learning model, and they can have a significant impact on model accuracy. If the hyperparameters are not set properly, the model may perform poorly, leading to low accuracy.
7. Poor data preprocessing: Data preprocessing is the process of cleaning, transforming, and preparing the data for machine learning. If the data is not preprocessed properly, it may contain noise or errors that can negatively impact model accuracy.

Overall, there are many potential causes of low accuracy when training a machine learning model, and it is often necessary to carefully analyze the data, model, and training process in order to identify and address any issues that may be contributing to low accuracy.

Reasons and Solutions

- Insufficient Data:

Reason: The dataset may be too small or unrepresentative.

Solution: We could augment the dataset, we could collect more diverse data, or use transfer learning.

- Overfitting:

Reason: The model is too complex and has memorized the training data, failing to generalize.

Solution: We could regularize the model (e.g., dropout, weight decay), reduce model complexity, or increase data augmentation.

- Underfitting:

Reason: The model is too simple and cannot capture the underlying patterns in the data.

Solution: Increase model complexity, add more features, or train for more epochs.

- Incorrect Model Architecture:

Reason: The chosen model architecture may not be suitable for the task.

YİĞİT SARIOĞLU
2022400354

Solution: Experiment with different architectures, consider state-of-the-art models, or use model ensembles.

- Poor Hyperparameter Tuning:

Reason: Suboptimal hyperparameter values (learning rate, batch size, etc.).

Solution: we could conduct hyperparameter tuning using techniques like grid search or random search.

- Data Quality Issues:

Reason: Noisy or incorrect labels, outliers, or missing data.

Solution: Clean and preprocess data, handle outliers, and ensure high-quality annotations.

- Imbalance in Data Distribution:

Reason: Uneven distribution of classes in the dataset.

Solution: Use techniques like oversampling, undersampling, or class-weighted loss functions.

- Inadequate Feature Engineering:

Reason: Relevant features might not be adequately represented.

Solution: Engineer new features, perform feature scaling, or consider domain-specific feature transformations.

- Learning Rate Issues:

Reason: Too high or too low learning rates can hinder convergence.

Solution: Experiment with different learning rates and use learning rate schedules.

- Lack of Regularization:

Reason: Insufficient regularization may lead to overfitting.

Solution: Introduce regularization techniques such as L1 or L2 regularization.

YİĞİT SARIOĞLU
2022400354

2. Strategies to Improve Accuracy

a. Adjust Hyperparameter Search Space

Based on the observations from the WandB dashboard, I adjusted hyperparameter search space. I have refined the range of values for each hyperparameter to focus on the most promising regions.

When I checked the results, I saw that this sweep/configuration had approximately 99% percent training accuracy, it suggests that the model is learning well on the training data. Also its validation accuracy is 73% , this is a high value compared to my other results. So I chose my ranges accordingly. To the solution , I've reduced the range for `drop_rate`, `l2_reg`, and `learning_rate` to focus the search on more promising values.

autumn-sweep-6 at: https://wandb.ai/bounteam/CMPE49T/runs/j3gwlimx

Key	Value
activation	"leaky_relu"
data_dir	"/tmp/curated_data/data/"
drop_rate	0.5900658457824832
early_stopping	15
image_size	128
l2_reg	0.03413657590244976
learning_rate	0.0684466354370893
nb_epoch	50
optimizer	"SGD"
test_batch_size	1
train_batch_size	64
val_batch_size	32

Key	Value
train_acc	0.993831391363948
train_loss	0.07259038437105952
val_acc	0.7305625223934074
val_loss	0.6350751861753706

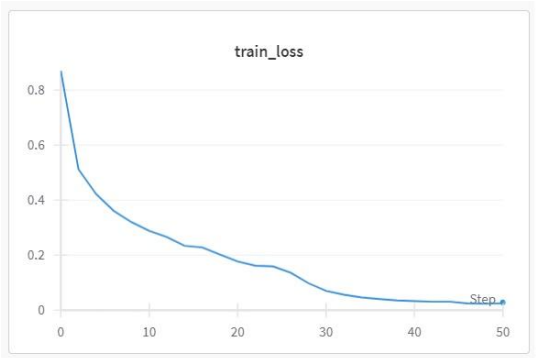
`I have changed the parameter-dict,` I've adjusted the ranges for `drop_rate`, `l2_reg`, and `learning_rate` to be more focused on the values that seem to work well.

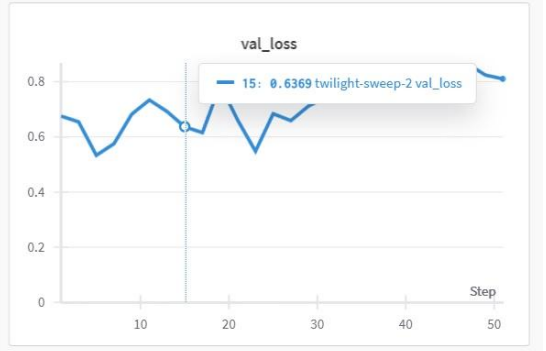
YİĞİT SARIOĞLU
2022400354

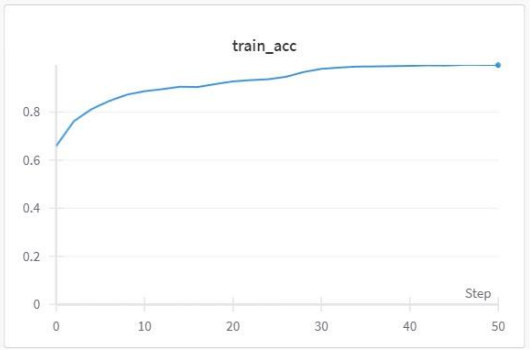
```
parameter_dict = {
    'activation': {
        'values': ['leaky_relu']
    },
    'optimizer': {
        'values': ['SGD']
    },
    'drop_rate': {
        'distribution': 'uniform',
        'min': 0.5,  # Narrowed down the range
        'max': 0.9
    },
    'l2_reg': {
        'distribution': 'uniform',
        'min': 0.02,  # Narrowed down the range
        'max': 0.05
    },
    'learning_rate': {
        'distribution': 'uniform',
        'min': 0.05,  # Narrowed down the range
        'max': 0.1
    }
}
```

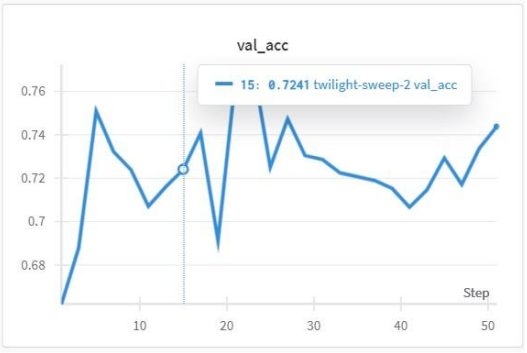
As a result of the adjustment on hyperparameters, I reached the followings: validation accuracy increased slightly and train loss decreased. [Syncing run **twilight-sweep-2**]

YİĞİT SARIOĞLU
2022400354









Config

View raw data

Config parameters describe your model's inputs. Learn more

Search keys

Key	Value
activation	"leaky_relu"
data_dir	"/tmp/curated_data/data/"
drop_rate	0.7936483813027613
early_stopping	15
image_size	128
l2_reg	0.020742987475364596
learning_rate	0.0814385274884265
nb_epoch	50
optimizer	"SGD"
test_batch_size	1
train_batch_size	64
val_batch_size	32

Summary

View raw data

Summary metrics describe your results. Learn more

Search keys

Key	Value
test_acc	"acc"
test_loss	"loss"
train_acc	0.9956591272561116
train_loss	0.028063530529627307
val_acc	0.7438194195628807
val_loss	0.8101935891086848

Comments on the first approach (hyperparameter selections / adjustments)

Results indicate that the hyperparameter adjustment had a positive impact on the validation accuracy. Here's a brief interpretation:

- Before Hyperparameter Adjustment:
  - Training Accuracy: 0.99
  - Validation Accuracy: 0.73
- After Hyperparameter Adjustment:
  - Training Accuracy: 0.99
  - Validation Accuracy: 0.74

YİĞİT SARIOĞLU
2022400354

In both cases, my model achieves very high training accuracy, indicating that it has learned the training data well. However, the key improvement is observed in the validation accuracy, which increased from 0.73 to 0.74 after the hyperparameter adjustment. It's important to note that achieving a perfect match between training and validation accuracy is not always desirable, as it might indicate overfitting. The goal is to strike a balance and ensure that the model generalizes well to new, unseen data.

b. Different CNN Architecture (experimenting with different architectures)

Potentially exploring different model architectures or data augmentation techniques, could lead to additional improvements or insights. I want to experiment with a slightly different architecture while keeping the basic structure similar to ResNet18. I want to increase the number of filters in each layer, add more convolutional blocks and then monitor validation performance. So I have changed the code slightly in the model architecture part.

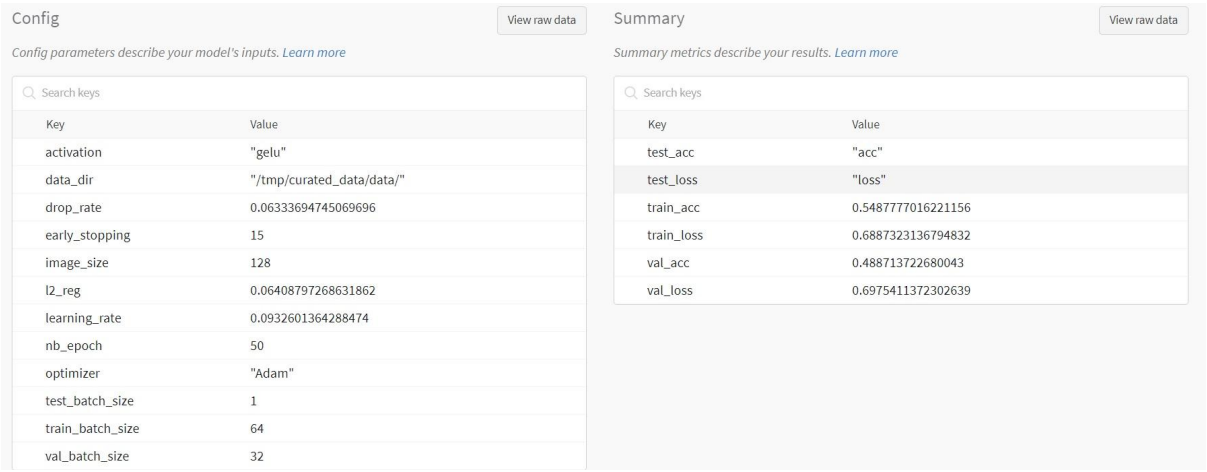
Modifications what i made : increases the number of filters in each convolutional layer, potentially allowing the model to capture more complex patterns in the data.

```
# This modification increases the number of filters in each
convolutional layer, potentially allowing the model to capture more
complex patterns in the data.
class ResNet18(nn.Module) :
    def __init__(self, act, drop_rate, image_size):
        super(ResNet18, self).__init__()
        if act == "relu":
            self.act_layer = nn.ReLU()
        elif act == "leaky_relu":
            self.act_layer = nn.LeakyReLU()
        elif act == "gelu":
            self.act_layer = nn.GELU()

        self.image_size = image_size

        self.conv1 = nn.Conv2d(1, 128, kernel_size=7, stride=2,
padding=3)  # 128 x N/2 x N/2
        self.bn1 = nn.BatchNorm2d(128)

        self.mp = nn.MaxPool2d((3, 3), stride=2, padding=1)  # 128 x
N/4 x N/4
```

```
        self.conv2a = ConvBlock(128, 128, 1, act)
        self.conv2b = ConvBlock(128, 128, 1, act)  # 128 x N/4 x N/4

        self.conv3a = ConvBlock(128, 256, 2, act)
        self.conv3b = ConvBlock(256, 256, 1, act)  # 256 x N/8 x N/8

        self.conv4a = ConvBlock(256, 512, 2, act)
        self.conv4b = ConvBlock(512, 512, 1, act)  # 512 x N/16 x N/16

        self.conv5a = ConvBlock(512, 1024, 2, act)
        self.conv5b = ConvBlock(1024, 1024, 1, act)  # 1024 x N/32 x
N/32

        kernel_size = self.image_size / (2 ** 5)
        self.avgpool = nn.AvgPool2d((int(kernel_size),
int(kernel_size)))
        self.flat = nn.Flatten()
        self.dropout = nn.Dropout(drop_rate)
        self.fc = nn.Linear(1024, 1)
```

Comments on the second approach (CNN architecture)

I used “gelu activation and “adam” optimizer in the first try and observed that train accuracy and validation accuracy dropped significantly. Likewise, I got low results (low training and validation accuracy) with the "leaky_relu" activation function and the "RMSprop" optimizer.



Config

View raw data

Config parameters describe your model's inputs. Learn more

Search keys

Key	Value
activation	"gelu"
data_dir	"/tmp/curated_data/data/"
drop_rate	0.06333694745069696
early_stopping	15
image_size	128
l2_reg	0.06408797268631862
learning_rate	0.0932601364288474
nb_epoch	50
optimizer	"Adam"
test_batch_size	1
train_batch_size	64
val_batch_size	32

Summary

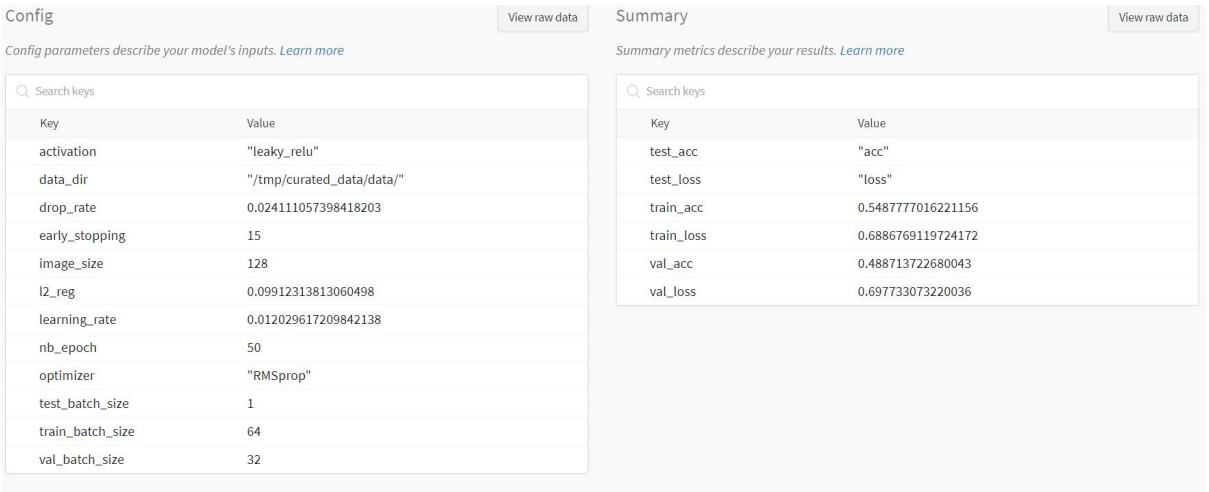
View raw data

Summary metrics describe your results. Learn more

Search keys

Key	Value
test_acc	"acc"
test_loss	"loss"
train_acc	0.5487777016221156
train_loss	0.6887323136794832
val_acc	0.488713722680043
val_loss	0.6975411372302639

YİĞİT SARIOĞLU
2022400354



Config

View raw data

Config parameters describe your model's inputs. Learn more

Search keys

Key	Value
activation	"leaky_relu"
data_dir	"/tmp/curated_data/data/"
drop_rate	0.024111057398418203
early_stopping	15
image_size	128
l2_reg	0.09912313813060498
learning_rate	0.012029617209842138
nb_epoch	50
optimizer	"RMSprop"
test_batch_size	1
train_batch_size	64
val_batch_size	32

Summary

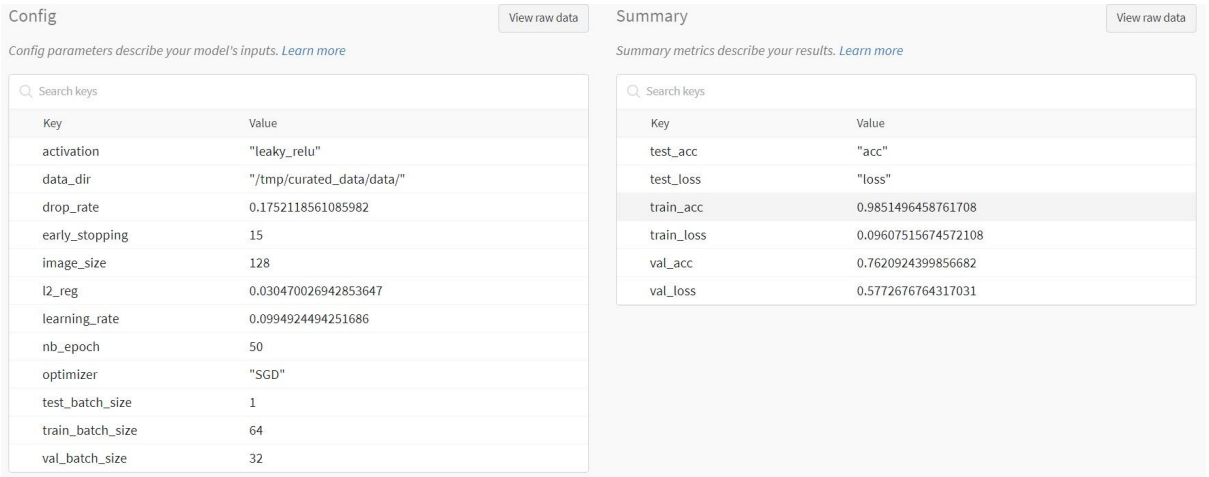
View raw data

Summary metrics describe your results. Learn more

Search keys

Key	Value
test_acc	"acc"
test_loss	"loss"
train_acc	0.5487777016221156
train_loss	0.6886769119724172
val_acc	0.488713722680043
val_loss	0.697733073220036

A configuration that yielded relatively better results was as follows:
- “leaky_relu” activation function and “SGD” optimizer parameters selected.
This still showed a decrease in training accuracy when compared to the ResNet model before the modification in the model. I also observed more training loss. There is a still small increase in validation accuracy.



Config

View raw data

Config parameters describe your model's inputs. Learn more

Search keys

Key	Value
activation	"leaky_relu"
data_dir	"/tmp/curated_data/data/"
drop_rate	0.1752118561085982
early_stopping	15
image_size	128
l2_reg	0.030470026942853647
learning_rate	0.0994924494251686
nb_epoch	50
optimizer	"SGD"
test_batch_size	1
train_batch_size	64
val_batch_size	32

Summary

View raw data

Summary metrics describe your results. Learn more

Search keys

Key	Value
test_acc	"acc"
test_loss	"loss"
train_acc	0.9851496458761708
train_loss	0.09607515674572108
val_acc	0.7620924399856682
val_loss	0.5772676764317031

3. Comments on Different Approximations

If there were sufficient GPU resources, we could experiment with more complex and deeper neural network architectures, as well as larger batch sizes. If GPU memory allows, we could try increasing the batch size. Larger batches may provide a more stable and accurate estimation of gradients.

The hyperparameter adjustment had a positive impact on the validation accuracy. Model achieves very high training accuracy (before and after the hyperparameter adjustment) , indicating that it has learned the training data well. However, the key improvement is observed in the validation accuracy, which increased from 0.73 to 0.74 after the hyperparameter adjustment.A validation accuracy of 0.74 suggests that the model is performing relatively well on unseen

data, and the adjustments made to hyperparameters have contributed to generalizing better on the validation set.

In the second approach, I have changed the architecture of the ResNet model slightly.I have increased the number of filters in each layer, add more convolutional blocks. The change in the model decreased the training accuracy when compared to the ResNet model before the modification.(also more training loss) . But there was a small increase in validation accuracy.

4. References

https://towardsdatascience.com/how-to-increase-the-accuracy-of-a-neural-network-9f5d1c6f407d

https://medium.com/@dnyaneshwalwadkar/fix-training-accuracy-fluctuation-over-fitting-problem-in-deep-learning-algorithm-859573090809

https://www.quora.com/What-is-the-number-one-cause-of-low-accuracy-when-training-a-machine-learning-model