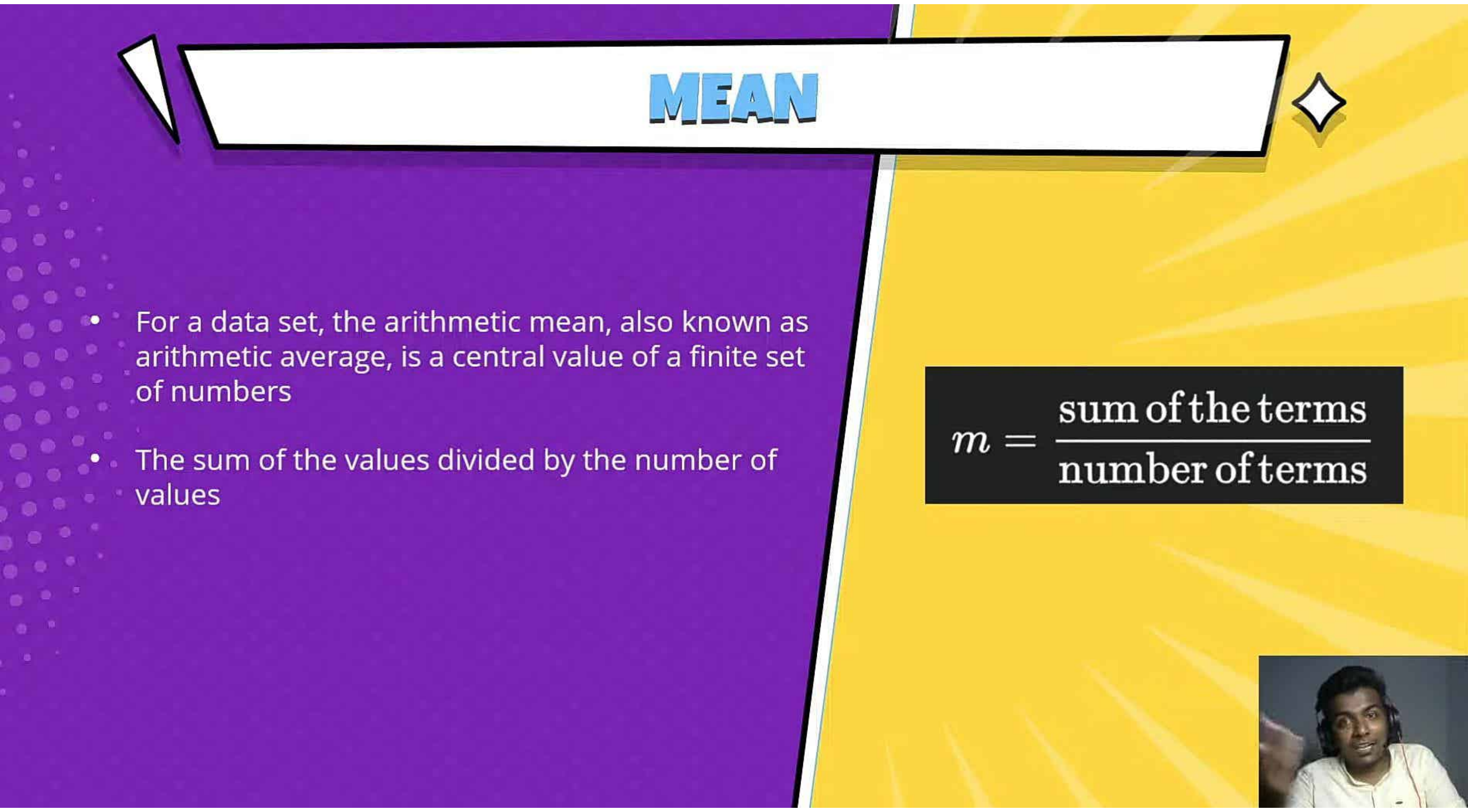
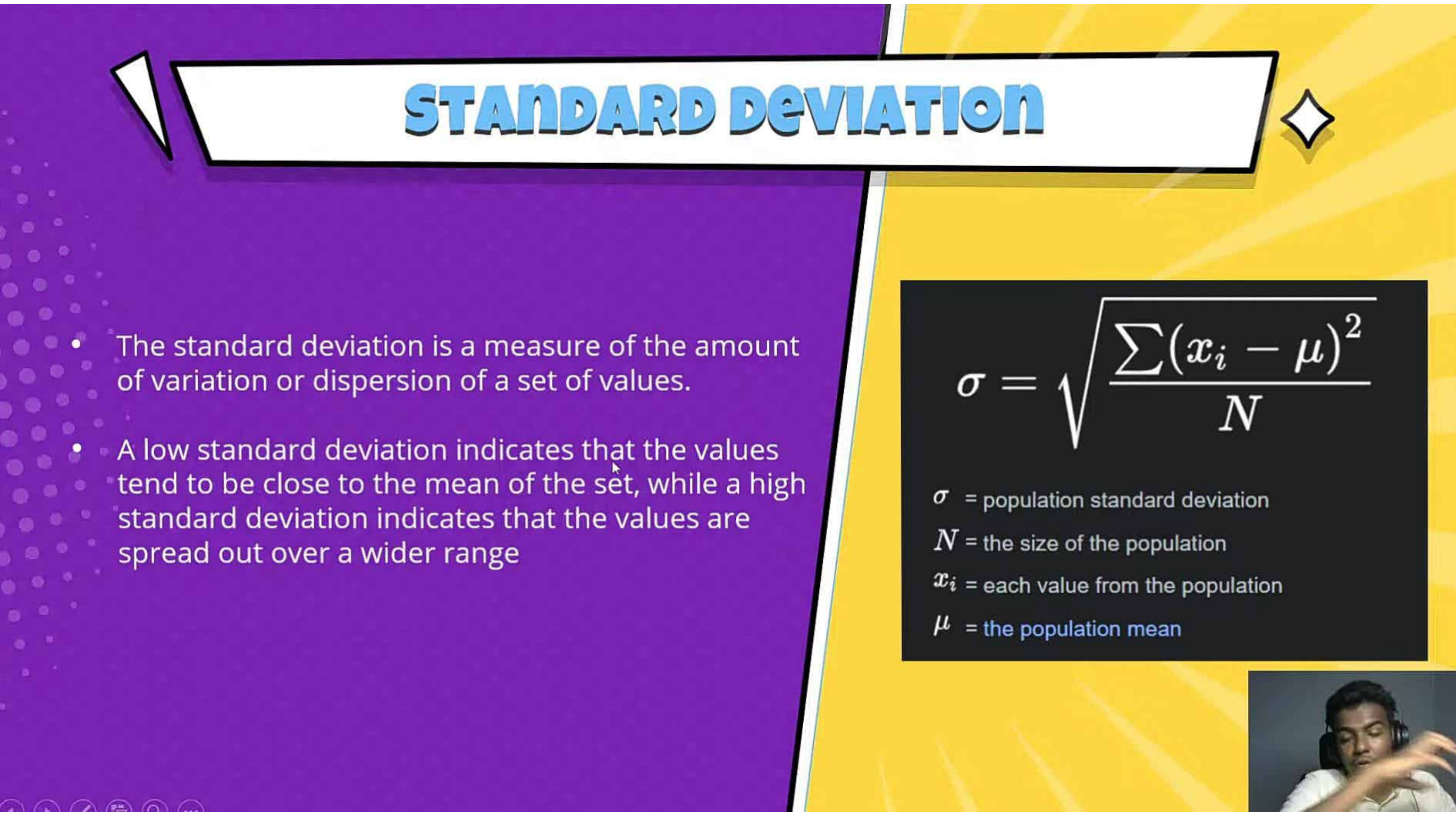


MEAN

- For a data set, the arithmetic mean, also known as arithmetic average, is a central value of a finite set of numbers
- The sum of the values divided by the number of values

$$m = \frac{\text{sum of the terms}}{\text{number of terms}}$$



STANDARD DEVIATION

- The standard deviation is a measure of the amount of variation or dispersion of a set of values.
- A low standard deviation indicates that the values tend to be close to the mean of the set, while a high standard deviation indicates that the values are spread out over a wider range

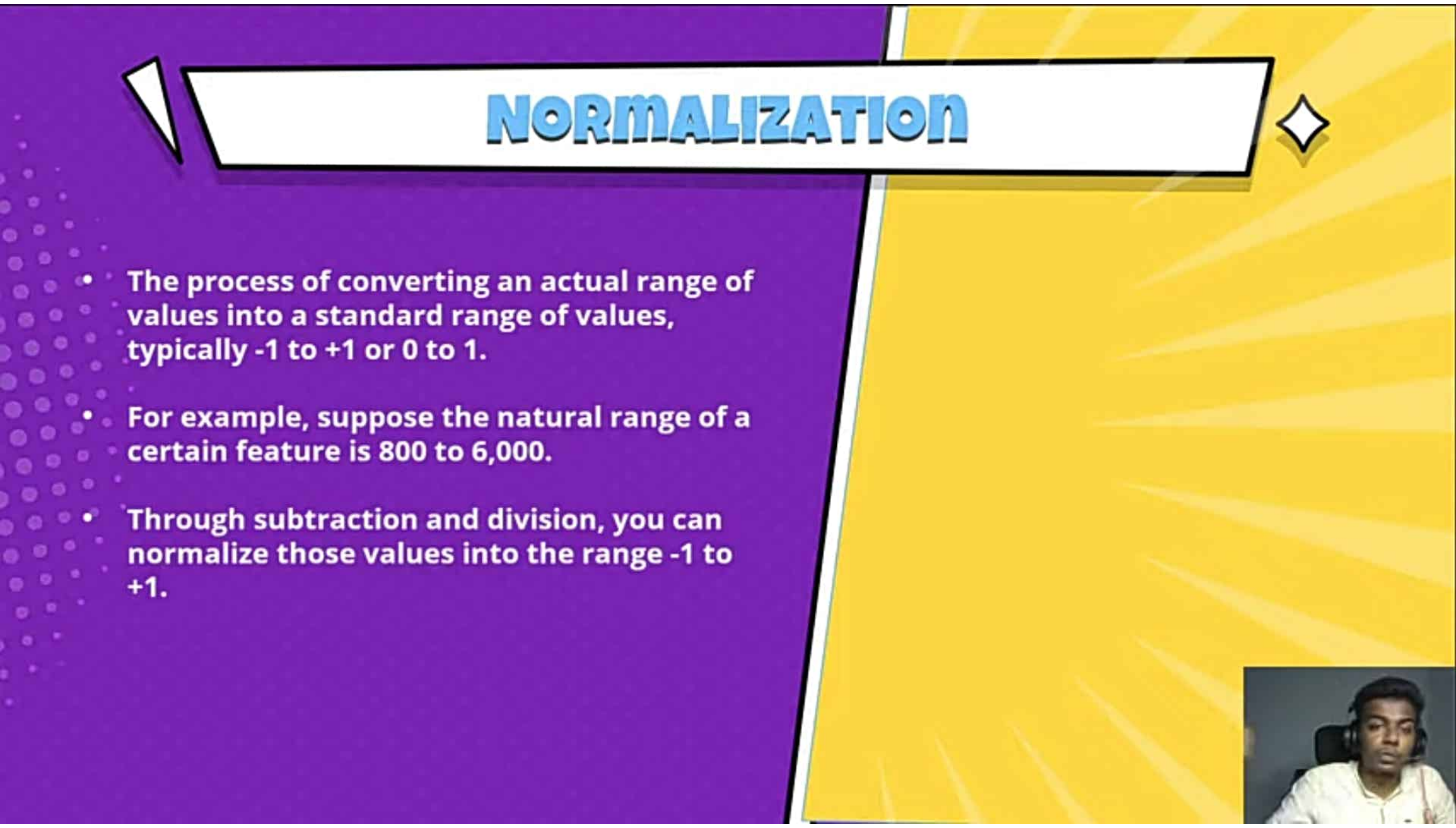
$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

σ = population standard deviation

N = the size of the population

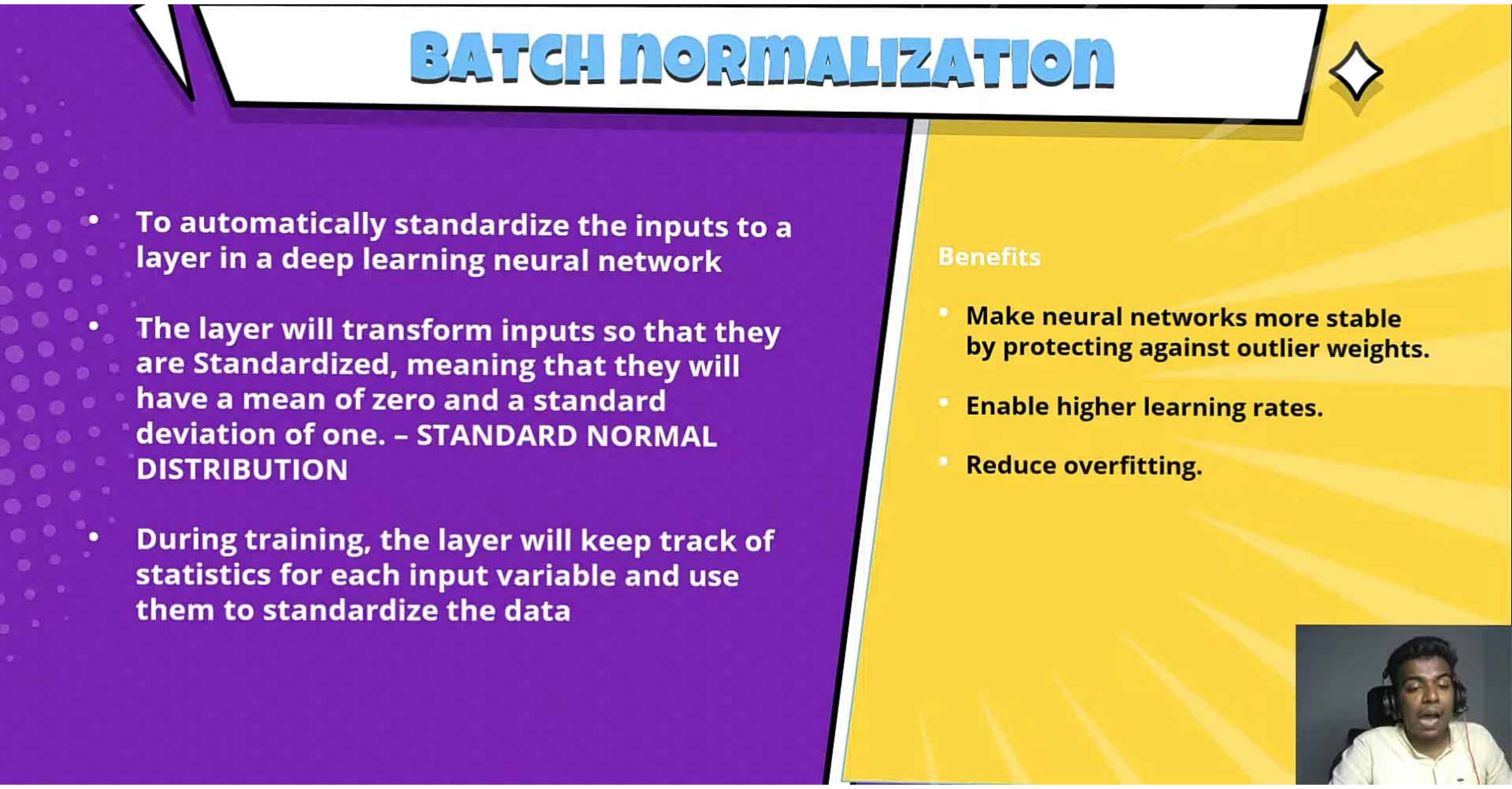
x_i = each value from the population

μ = the population mean



NORMALIZATION

- The process of converting an actual range of values into a standard range of values, typically -1 to +1 or 0 to 1.
- For example, suppose the natural range of a certain feature is 800 to 6,000.
- Through subtraction and division, you can normalize those values into the range -1 to +1.

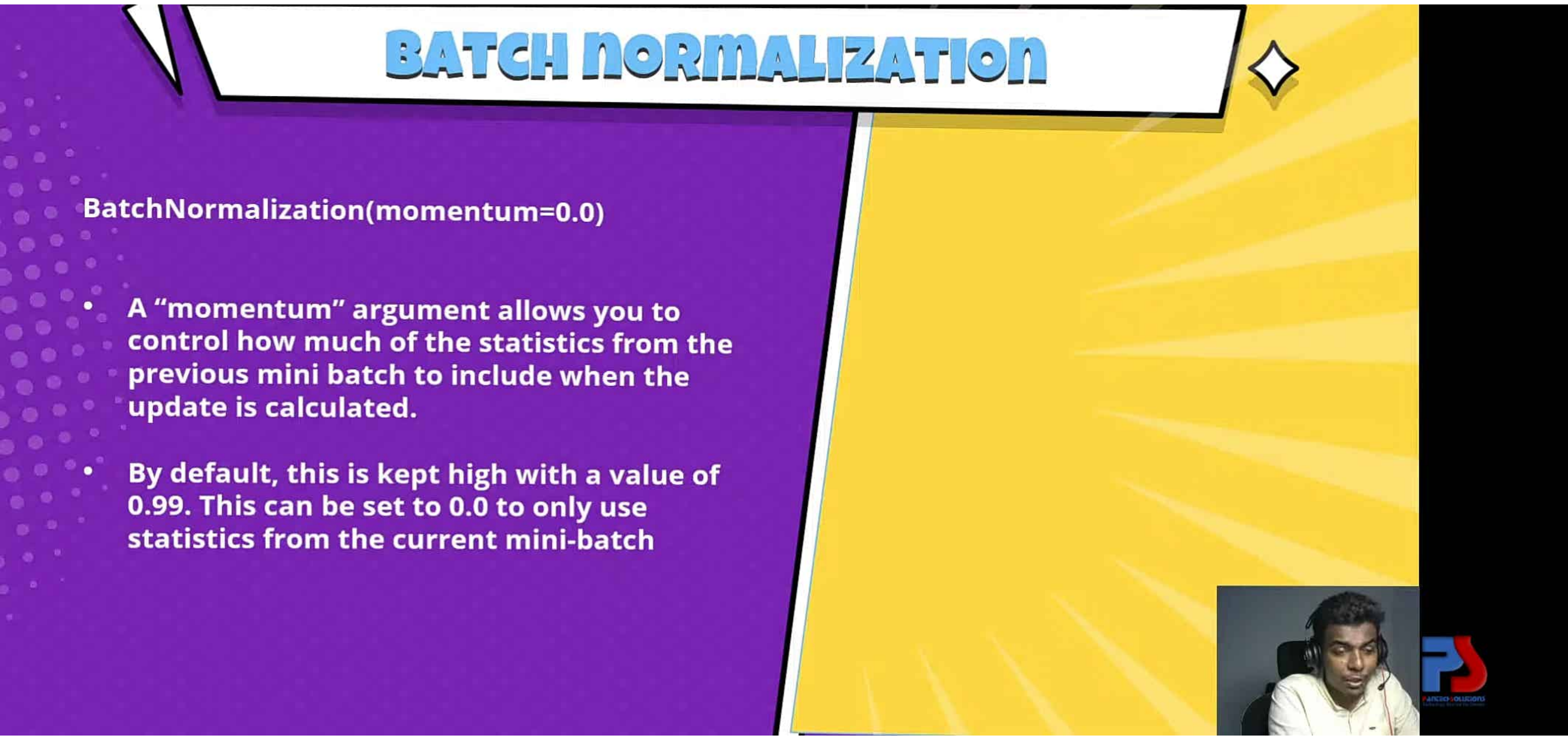


BATCH NORMALIZATION

- To automatically standardize the inputs to a layer in a deep learning neural network
- The layer will transform inputs so that they are Standardized, meaning that they will have a mean of zero and a standard deviation of one. – STANDARD NORMAL DISTRIBUTION
- During training, the layer will keep track of statistics for each input variable and use them to standardize the data

Benefits

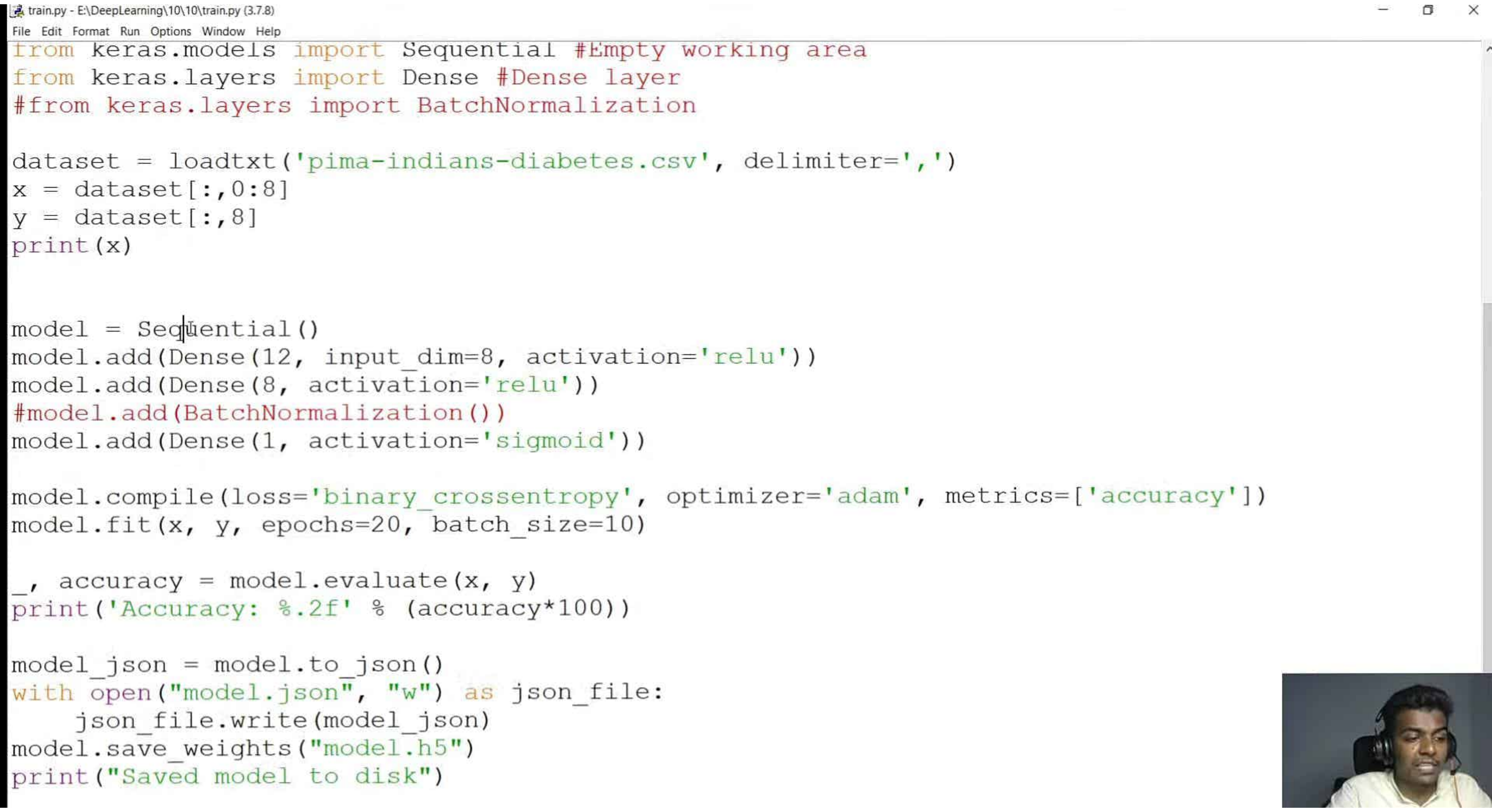
- Make neural networks more stable by protecting against outlier weights.
- Enable higher learning rates.
- Reduce overfitting.



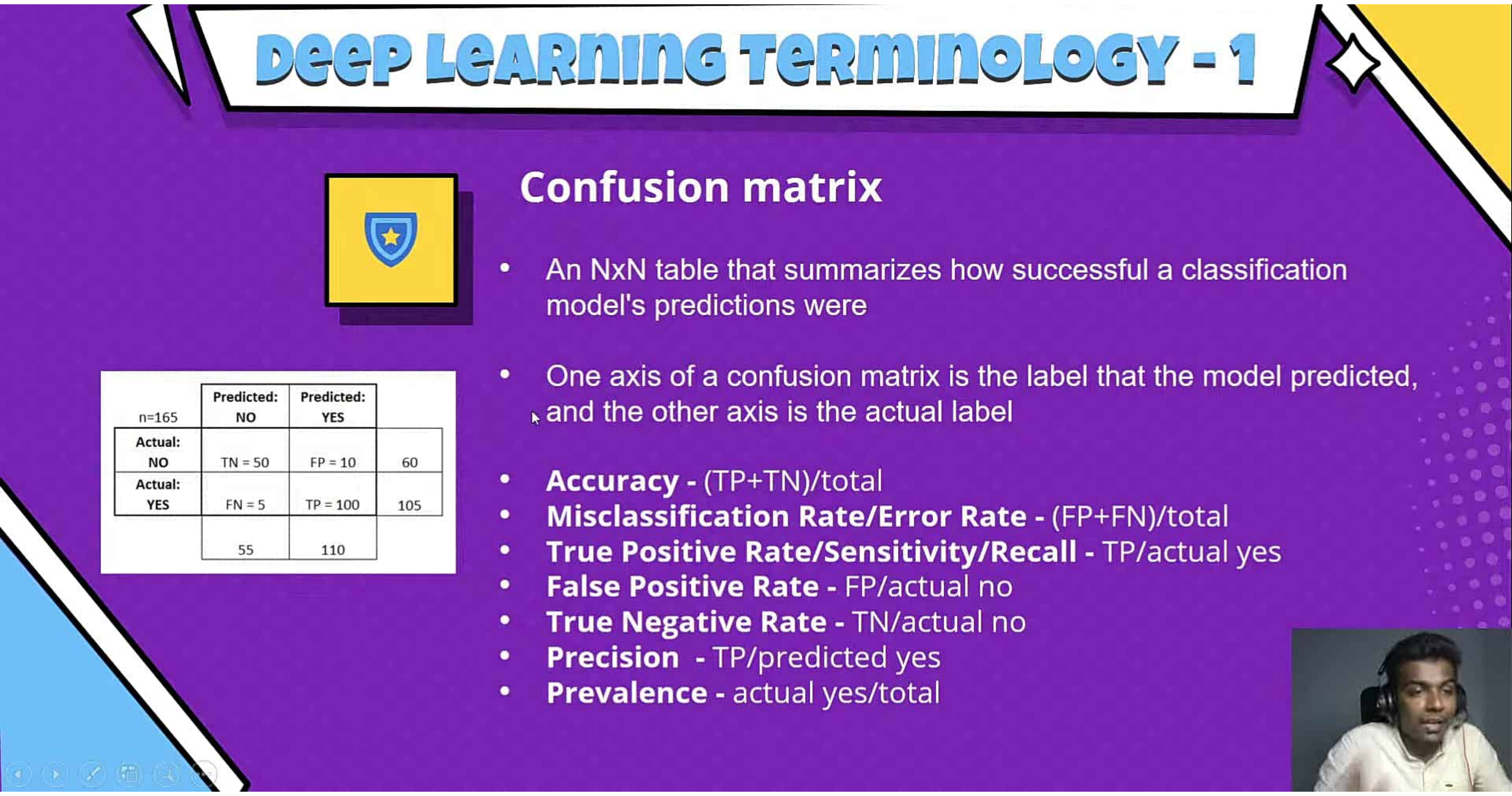
BATCH NORMALIZATION

BatchNormalization(momentum=0.0)

- A “momentum” argument allows you to control how much of the statistics from the previous mini batch to include when the update is calculated.
- By default, this is kept high with a value of 0.99. This can be set to 0.0 to only use statistics from the current mini-batch


train.py - E:\DeepLearning\10\10\train.py (3.7.8)
File Edit Format Run Options Window Help

```
from keras.models import Sequential #Empty working area
from keras.layers import Dense #Dense layer
#from keras.layers import BatchNormalization

dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
x = dataset[:,0:8]
y = dataset[:,8]
print(x)


model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
#model.add(BatchNormalization())
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x, y, epochs=20, batch_size=10)

_, accuracy = model.evaluate(x, y)
print('Accuracy: %.2f' % (accuracy*100))

model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("model.h5")
print("Saved model to disk")
```



Deep Learning Terminology - 1

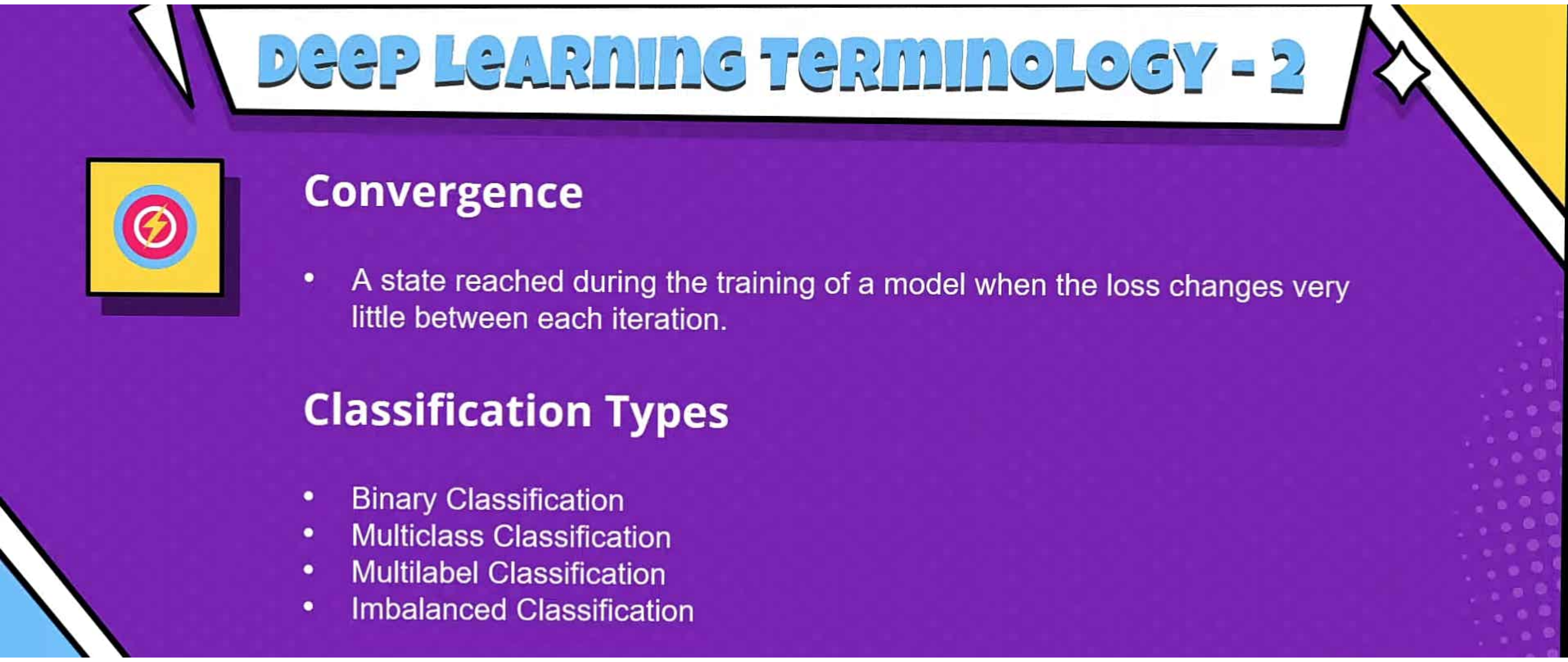
Confusion matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- An NxN table that summarizes how successful a classification model's predictions were

- One axis of a confusion matrix is the label that the model predicted, and the other axis is the actual label

- **Accuracy -** (TP+TN)/total
- **Misclassification Rate/Error Rate -** (FP+FN)/total
- **True Positive Rate/Sensitivity/Recall -** TP/actual yes
- **False Positive Rate -** FP/actual no
- **True Negative Rate -** TN/actual no
- **Precision -** TP/predicted yes
- **Prevalence -** actual yes/total



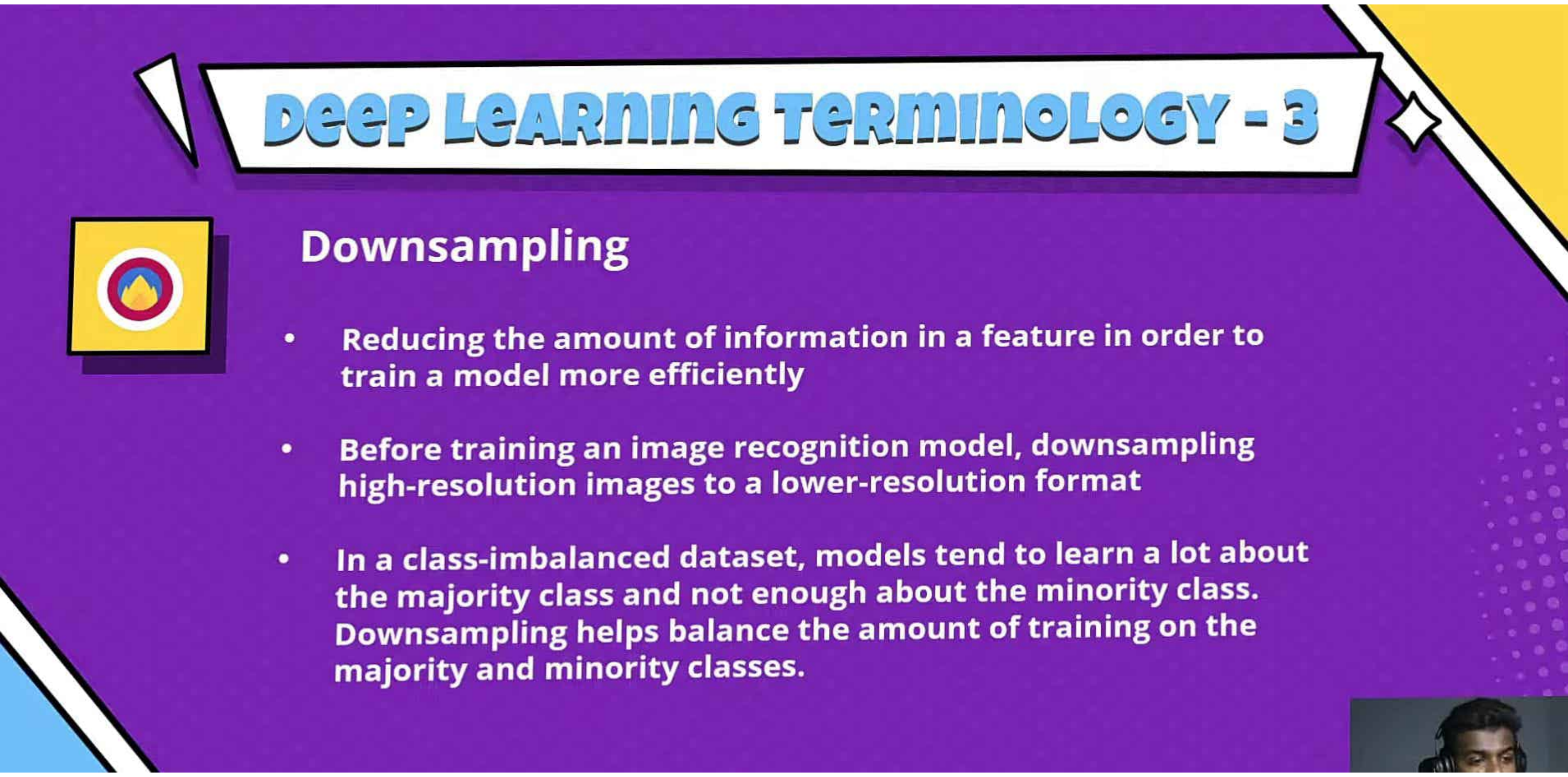
Deep Learning Terminology - 2

Convergence

- A state reached during the training of a model when the loss changes very little between each iteration.

Classification Types

- Binary Classification
- Multiclass Classification
- Multilabel Classification
- Imbalanced Classification



Deep Learning Terminology - 3

Downsampling

- Reducing the amount of information in a feature in order to train a model more efficiently
- Before training an image recognition model, downsampling high-resolution images to a lower-resolution format
- In a class-imbalanced dataset, models tend to learn a lot about the majority class and not enough about the minority class. Downsampling helps balance the amount of training on the majority and minority classes.