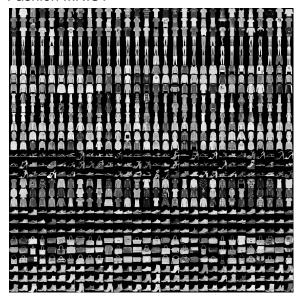
Laporan Project UAS Machine Learning

Kelompok:

Leonardo Marcellino Gunawan - C14210030 Jason Christopher - C14210098 Michael Putra Handidjaja - C14210184

Dataset

Fashion MNIST



Fashion-MNIST Dataset | Papers With Code

Fashion-MNIST is a dataset comprising of 28×28 grayscale images of 70,000 fashion products from 10 categories, with 7,000 images per category. The training set has 60,000 images and the test set has 10,000 images. Fashion-MNIST shares the same image size, data format and the structure of training and testing splits with the original MNIST.

Data Preprocessing

Check Null

```
if np.any(np.isnan(X_train)):
   print("Null values found in X train.")
else:
   print("No null values in X train.")
# Check for null values in y train
if np.any(np.isnan(y train)):
   print("Null values found in y train.")
else:
    print("No null values in y train.")
if np.any(np.isnan(X test)):
   print("Null values found in X test.")
else:
    print("No null values in X test.")
# Check for null values in y test
if np.any(np.isnan(y test)):
   print("Null values found in y test.")
else:
   print("No null values in y test.")
```

```
No null values in X_train.

No null values in y_train.

No null values in X_test.

No null values in y test.
```

Check the data distribution

```
classes = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
    'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

# Count the occurrences of each class in y_train
train_class_counts = np.bincount(y_train)

# Count the occurrences of each class in y_test
```

```
test_class_counts = np.bincount(y_test)

# Print the data distribution for y_train with labels
print("Data Distribution for y_train:")
for i in range(len(classes)):
    print(f'{classes[i]}: {train_class_counts[i]}')

# Print the data distribution for y_test with labels
print("\nData Distribution for y_test:")
for i in range(len(classes)):
    print(f'{classes[i]}: {test_class_counts[i]}')
```

```
Data Distribution for y train:
T-shirt/top: 6000
Trouser: 6000
Pullover: 6000
Dress: 6000
Coat: 6000
Sandal: 6000
Shirt: 6000
Sneaker: 6000
Baq: 6000
Ankle boot: 6000
Data Distribution for y test:
T-shirt/top: 1000
Trouser: 1000
Pullover: 1000
Dress: 1000
Coat: 1000
Sandal: 1000
Shirt: 1000
Sneaker: 1000
Baq: 1000
Ankle boot: 1000
```

Shuffle dan Normalize data supaya range nya menjadi 0-1

```
from sklearn.utils import shuffle

# Shuffle the training data

X_train, y_train = shuffle(X_train, y_train, random_state=42)

X_test, y_test = shuffle(X_test, y_test, random_state=42)

X_train = X_train / 255

X_test = X_test / 255
```

Metode CNN

Metode CNN menggunakan AlexNet architecture dengan adaptive learning rate.

Parameter Configuration

Configuration ini dijalankan dengan 15 epochs semua dan menggunakan loss sparse_categorical_crossentropy karena output datanya berupa angka 0-9 (yang merupakan indikasi dari class-classnya)

Configurati on	Layers	Regularization	Optimizer	Train Error	Test Error
Conf1	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1 4. Convolutional Layer 2: - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same' 5. Batch Normalization Layer 2: - Applies batch normalization after Convolutional Layer 2. 6. Max Pooling Layer 2: - Pool Size: (3, 3) - Strides: 1	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.2498	0.2757

4, 5			
	Convolutional Layer 3: - Filters: 64		
	- Kernel Size: (5, 5) - Activation: ReLU		
-	- Padding: 'same'		
-	Convolutional Layer 4: - Filters: 64		
	- Kernel Size: (5, 5) - Activation: ReLU		
	- Padding: 'same' Convolutional Layer 5:		
-	- Filters: 32 - Kernel Size: (5, 5)		
-	- Activation: ReLU		
	- Padding: 'same'		
	Max Pooling Layer 3: Pool Size: (3, 3)		
- S	Strides: 1		
	latten Layer: Flattens the input to		
prep	oare for the fully		
	nected layers.		
	Dense Layer 1: Units: 64		
	Activation: ReLU Weight Initialization:		
Glo	rotUniform Batch Normalization		
-1	Dense Layer 2: Units: 64		
	Activation: ReLU Weight Initialization:		
	rotUniform Batch Normalization		
	Output Dense Layer:		
- 1	Units: 10 (for 10		
-,	Activation: Softmax ´		
	Weight Initialization: rotUniform		
clas	sses in classification) Activation: Softmax Weight Initialization:		

	T		ı	I	
Conf2	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1 4. Convolutional Layer 2: - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same' 5. Batch Normalization Layer 2: - Applies batch normalization after Convolutional Layer 2: - Pool Size: (3, 3) - Strides: 1 7. Convolutional Layer 3: - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same' - Convolutional Layer 4: - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same' - Convolutional Layer 5: - Filters: 32 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same' - Convolutional Layer 5: - Filters: 32 - Kernel Size: (5, 5) - Activation: ReLU	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9		

- Padding: 'same' 8. Max Pooling Layer 3: - Pool Size: (3, 3) - Strides: 1 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform 12. Conf3 1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1 4. Convolutional Layer 2:						I
- Pool Size: (3, 3) - Strides: 1 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform 1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1: - Applies batch normalization after Convolutional Layer 1: - Pool Size: (3, 3) - Strides: 1		- Padding: 'same'				
- Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform 1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1		- Pool Size: (3, 3)				
- Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform 11. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1: - Pool Size: (3, 3) - Strides: 1		- Flattens the input to prepare for the fully				
- Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform 1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1 Pool Size: (3, 3) - Strides: 1		- Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform				
- Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform 1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1: - Applies batch normalization after Convolutional Layer 1: - Pool Size: (3, 3) - Strides: 1		- Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform				
Conf3 - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1 Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42		- Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization:				
2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1	Conf3	- Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU	Layer 1: - Rate: 0.25 - Seed: 42	momentum	0.2352	0.2714
- Pool Size: (3, 3) - Strides: 1		Layer 1: - Applies batch normalization after	Layer 2: - Rate: 0.25			
4. Convolutional Laver 2.		- Pool Size: (3, 3)				
555.25.2.		4. Convolutional Layer 2:				

,			
- Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same'			
5. Batch Normalization Layer 2: - Applies batch normalization after Convolutional Layer 2.			
6. Max Pooling Layer 2: - Pool Size: (3, 3) - Strides: 1			
7. Convolutional Layers 4, 5: - Convolutional Layer - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same' - Convolutional Layer - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same' - Convolutional Layer - Filters: 32 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same'	3: 4:		
8. Max Pooling Layer 3: - Pool Size: (3, 3) - Strides: 1			
9. Flatten Layer: - Flattens the input to prepare for the fully connected layers.			
10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization			
11. Dense Layer 2:			

	- Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf4	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1 4. Convolutional Layer 2: - Filters: 64 - Kernel Size: (7, 7) - Activation: ReLU - Padding: 'same' 5. Batch Normalization Layer 2: - Applies batch normalization after Convolutional Layer 2: - Pool Size: (5, 5) - Strides: 2 7. Convolutional Layer 3: - Filters: 64 - Kernel Size: (3, 3)	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.3298	0.3129

	- Activation: ReLU - Padding: 'same' - Convolutional Layer 4: - Filters: 64 - Kernel Size: (3, 3) - Activation: ReLU - Padding: 'same' - Convolutional Layer 5: - Filters: 32 - Kernel Size: (3, 3) - Activation: ReLU - Padding: 'same' 8. Max Pooling Layer 3: - Pool Size: (3, 3) - Strides: 1 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization:				
	- Weight Initialization: GlorotUniform				
Conf5	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (11, 11) - Activation: ReLU - Input Shape: (28, 28, 1)	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.2093	0.2816
	2. Batch Normalization	2. Dropout Layer 2:			

Layer 1: - Applies batch normalization after Convolutional Layer 1.	- Rate: 0.25 - Seed: 42		
3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 2			
4. Convolutional Layer 2: - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same'			
5. Batch Normalization Layer 2: - Applies batch normalization after Convolutional Layer 2.			
6. Max Pooling Layer 2: - Pool Size: (3, 3) - Strides: 2			
7. Convolutional Layers 3, 4, 5: - Convolutional Layer 3: - Filters: 64 - Kernel Size: (3, 3) - Activation: ReLU - Padding: 'same' - Convolutional Layer 4: - Filters: 64 - Kernel Size: (3, 3) - Activation: ReLU - Padding: 'same' - Convolutional Layer 5: - Filters: 32 - Kernel Size: (3, 3) - Activation: ReLU - Padding: 'same'			
8. Max Pooling Layer 3: - Pool Size: (3, 3) - Strides: 2			
9. Flatten Layer: - Flattens the input to prepare for the fully			

	T	<u> </u>			Ι
	connected layers. 10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf6	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (4, 4) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (2, 2) - Strides: 1 4. Convolutional Layer 2: - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Padding: 'same' 5. Batch Normalization Layer 2: - Applies batch normalization after Convolutional Layer 2.	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.1549	0.2672

г г			
	 6. Max Pooling Layer 2: Pool Size: (2,2)Strides: 2 7. Convolutional Layers 3, 4, 5:		
	 Convolutional Layer 3: Filters: 64 Kernel Size: (3, 3) Activation: ReLU Padding: 'same' Convolutional Layer 4: Filters: 64 Kernel Size: (3, 3) Activation: ReLU Padding: 'same' Convolutional Layer 5: Filters: 32 		
	- Kernel Size: (3, 3)- Activation: ReLU- Padding: 'same'		
	8. Max Pooling Layer 3: - Pool Size: (2, 2) - Strides: 1		
	9. Flatten Layer: - Flattens the input to prepare for the fully connected layers.		
	10. Dense Layer 1: - Units: 128 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization		
	11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization		
	12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax		

	- Weight Initialization: GlorotUniform				
Conf7	1. Convolutional Layer 1: Filters: 32 Kernel Size: (5, 5) Activation: ReLU Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: Pool Size: (5, 5) Strides: 2 4. Convolutional Layer 2: Filters: 64 Kernel Size: (5, 5) Activation: ReLU Padding: 'same' 5. Batch Normalization Layer 2: Applies batch normalization after Convolutional Layer 2: Pool Size: (5,5) Strides: 2 7. Convolutional Layer 2: Pool Size: (5,5) Strides: 2 7. Convolutional Layer 3: Filters: 64 Kernel Size: (5, 5) Activation: ReLU Padding: 'same' Convolutional Layer 4: Filters: 64 Kernel Size: (5, 5) Activation: ReLU Padding: 'same' Convolutional Layer 5: Filters: 32 Kernel Size: (5, 5)	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.1057	0.2574

	- Activation: ReLU - Padding: 'same' 8. Max Pooling Layer 3: - Pool Size: (2, 2) - Strides: 2 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 128 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64				
	- Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization				
	12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf8	1. Convolutional Layer 1: - Filters: 64 - Kernel Size: (5, 5) - Activation: ReLU - Input Shape: (28, 28, 1)	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.194	0.2481
	2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1.				
	3. Max Pooling Layer 1: - Pool Size: (3, 3)				

Strides: 1
1. Convolutional Layer 2:
Filters: 32
Kernel Size: (5, 5)
Activation: ReLU
Padding: 'same'
5. Batch Normalization
.ayer 2:
Applies batch
normalization after
Convolutional Layer 2.
´
5. Max Pooling Layer 2:
Pool Size: (3, 3)
Strides: 2
7. Convolutional Layers 3, 4,
5:
Convolutional Layer 3:
Filters: 64
Kernel Size: (5, 5)
Activation: ReLU
Padding: 'same'
Convolutional Layer 4:
Filters: 64
Kernel Size: (5, 5)
Activation: ReLU
Padding: 'same'
Convolutional Layer 5:
Filters: 32
Kernel Size: (5, 5)
Activation: ReLU
Padding: 'same'
adding. Same
3. Max Pooling Layer 3:
Pool Size: (3, 3)
Strides: 1
Suiucs. I
) Flatten Laver:
9. Flatten Layer: Flattens the input to
riations the input to

	prepare for the fully connected layers. 10. Dense Layer 1: - Units: 128 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform			
Conf9	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (5, 5) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1 4. Convolutional Layer 2:	SGD momentum 0.9	0.2064	0.2511

- Filters: 32			
- Kernel Size: (5, 5)			
- Activation: ReLU			
- Padding: 'same'			
l adding came			
6. Max Pooling Layer 2:			
1			
- Pool Size: (3, 3)			
- Strides: 2			
	_		
7. Convolutional Layers	3, 4,		
5:			
- Convolutional Layer 3:			
- Filters: 32			
- Kernel Size: (5, 5)			
- Activation: ReLU			
- Padding: 'same'			
- Convolutional Layer 4:			
- Filters: 32			
- Kernel Size: (5, 5)			
- Activation: ReLU			
- Padding: 'same'			
- Convolutional Layer 5:			
<u> </u>			
- Filters: 32			
- Kernel Size: (5, 5)			
- Activation: ReLU			
- Padding: 'same'			
8. Max Pooling Layer 3:			
- Pool Size: (3, 3)			
- Strides: 1			
- Strides. I			
O Flatter Laver			
9. Flatten Layer:			
- Flattens the input to			
prepare for the fully			
connected layers.			
10 Dance Layer 1:			
10. Dense Layer 1:			
- Units: 64			
- Activation: ReLU			
- Weight Initialization:			
GlorotUniform			
- Batch Normalization			
!			

	11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf10	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (7, 7) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 2 4. Convolutional Layer 2: - Filters: 64 - Kernel Size: (7, 7) - Activation: ReLU - Padding: 'same' 5. Batch Normalization Layer 2: - Applies batch normalization after Convolutional Layer 2: - Pool Size: (5, 5) - Strides: 2 7. Convolutional Layers 3, 4, 5: - Convolutional Layer 3: - Filters: 64 - Kernel Size: (4, 4)	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.199	0.2776

	I				
	- Activation: ReLU - Padding: 'same' - Convolutional Layer 4: - Filters: 64 - Kernel Size: (4, 4) - Activation: ReLU - Padding: 'same' - Convolutional Layer 5: - Filters: 32 - Kernel Size: (4, 4) - Activation: ReLU - Padding: 'same' 8. Max Pooling Layer 3: - Pool Size: (3, 3) - Strides: 2 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf11	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (5, 5) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.1508	19.7609

	Γ		
Convolutional Layer 1.			
3. Max Pooling Layer 1:			
- Pool Size: (3, 3)			
- Strides: 1			
4. Comvolutional Layer 2:			
4. Convolutional Layer 2: - Filters: 32			
- Kernel Size: (5, 5) - Activation: ReLU			
- Padding: 'same'			
5. Batch Normalization			
Layer 2:			
Layer 2.			
- Applies batch			
normalization after			
Convolutional Layer 2.			
6. Max Pooling Layer 2:			
- Pool Size: (3, 3)			
- Strides: 1			
7. Convolutional Layers 3, 4,			
5:			
- Convolutional Layer 3:			
- Filters: 32			
- Kernel Size: (5, 5)			
- Activation: ReLU			
- Padding: 'same'			
- Convolutional Layer 4:			
- Filters: 32			
- Kernel Size: (5, 5)			
- Activation: ReLU			
- Padding: 'same'			
- Convolutional Layer 5:			
- Filters: 32			
- Kernel Size: (5, 5)			
- Activation: ReLU			
- Padding: 'same'			
8. Max Pooling Layer 3:			

	- Pool Size: (3, 3)				
	- Strides: 1 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers.				
	10. Dense Layer 1: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform				
	11. Dense Layer 2: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization				
	12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf12	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (5, 5) - Activation: ReLU - Input Shape: (28, 28, 1)	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.6004	0.439
	2. Batch NormalizationLayer 1:- Applies batchnormalization afterConvolutional Layer 1.				
	3. Max Pooling Layer 1:				

- Pool Size: (3, 3)		
- Strides: 1		
Gurado. 1		
4 Convolutional Lover 2:		
4. Convolutional Layer 2:		
- Filters: 32		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
-		
5. Batch Normalization		
Layer 2:		
-		
- Applies batch		
normalization after		
Convolutional Layer 2.		
6. Max Pooling Layer 2:		
- Pool Size: (3, 3)		
- Strides: 1		
7. Convolutional Layers 3, 4,		
5:		
- Convolutional Layer 3:		
- Filters: 32		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
- Convolutional Layer 4:		
- Filters: 32		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
- Convolutional Layer 5:		
- Filters: 32		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
8. Max Pooling Layer 3:		
- Pool Size: (3, 3)		
· · · ·		
- Strides: 1		
9. Flatten Layer:		

	- Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform 11. Dense Layer 2: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf13	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (5, 5) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 2 4. Convolutional Layer 2:	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.2044	0.2609

- Filters: 32		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
5. Batch Normalization		
Layer 2:		
- Applies batch		
normalization after		
Convolutional Layer 2.		
Convolutional Layer 2.		
. M. D. II. I. O.		
6. Max Pooling Layer 2:		
- Pool Size: (3, 3)		
- Strides: 2		
7. Convolutional Layers 3, 4,		
5:		
- Convolutional Layer 3:		
- Filters: 32		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
- Convolutional Layer 4:		
- Filters: 32		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
- Convolutional Layer 5:		
- Filters: 32		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
8. Max Pooling Layer 3:		
- Pool Size: (3, 3)		
- Strides: 2		
Guides. 2		
O Flotten Laver:		
9. Flatten Layer:		
- Flattens the input to		
prepare for the fully		
connected layers.		

	10. Dense Layer 1: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 10 (for 10 classes in classification)				
	- Activation: Softmax - Weight Initialization: GlorotUniform				
Conf14	1. Convolutional Layer 1: - Filters: 64 - Kernel Size: (4, 4) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (4, 4) - Strides: 2 4. Convolutional Layer 2: - Filters: 64 - Kernel Size: (4, 4) - Activation: ReLU - Padding: 'same' 5. Batch Normalization Layer 2: - Applies batch normalization after Convolutional Layer 2.	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.1937	0.2473

	6. Max Pooling Layer 2:		
	- Pool Size: (4, 4)		
	- Strides: 2		
	7. Convolutional Layers 3,		
	4, 5:		
	- Convolutional Layer 3: - Filters: 64		
	- Kernel Size: (4, 4)		
	- Activation: ReLU		
	- Padding: 'same'		
	- Convolutional Layer 4:		
	- Filters: 64		
	- Kernel Size: (4, 4)		
	- Activation: ReLU		
	- Padding: 'same'		
	- Convolutional Layer 5:		
	- Filters: 64		
	- Kernel Size: (4, 4)		
	- Activation: ReLU		
	- Padding: 'same'		
	8. Max Pooling Layer 3:		
	- Pool Size: (4, 4)		
	- Strides: 2		
	9. Flatten Layer:		
	- Flattens the input to		
	prepare for the fully		
	connected layers.		
	10. Dense Layer 1:		
	- Units: 64		
	- Activation: ReLU		
	- Weight Initialization:		
	GlorotUniform		
	- Batch Normalization		
	11. Dense Layer 2:		
	- Units: 32		
	- Activation: ReLU		
	- Weight Initialization:		
	GlorotUniform		
	- Batch Normalization		
	12. Output Dense Layer:		
	- Units: 10 (for 10 classes		
	in classification)		
	- Activation: Softmax		
	- Weight Initialization:		
	GlorotUniform		
·		 ·	

			000	0 1774	0.000
	1. Convolutional Layer 1:	1. Dropout	SGD	0.1774	0.2808
Conf15	- Filters: 32	Layer 1:	momentum		
	- Kernel Size: (10, 10)	- Rate: 0.25	0.9		
	- Activation: ReLU	- Seed: 42			
	- Input Shape: (28, 28, 1)				
		2. Dropout			
	2. Batch Normalization	Layer 2:			
	Layer 1:	- Rate: 0.25			
	- Applies batch	- Seed: 42			
	normalization after				
	Convolutional Layer 1.				
	3. Max Pooling Layer 1:				
	- Pool Size: (2, 2)				
	- Strides: 2				
	Strides. 2				
	4. Convolutional Layer 2:				
	- Filters: 64				
	- Kernel Size: (5, 5)				
	- Activation: ReLU				
	- Padding: 'same'				
	5 5 . 1 . 1				
	5. Batch Normalization				
	Layer 2:				
	- Applies batch				
	normalization after				
	Convolutional Layer 2.				
	6. Max Pooling Layer 2:				
	- Pool Size: (2, 2)				
	- Strides: 2				
	7. Convolutional Layers 3, 4,				
	5:				
	- Convolutional Layer 3:				
	- Filters: 64				
	- Kernel Size: (3, 3)				
	- Activation: ReLU				
	- Padding: 'same'				
	- Convolutional Layer 4:				
	- Filters: 64				
	- Kernel Size: (3, 3)				
	1.011101 0120. (0, 0)				

	- Activation: ReLU - Padding: 'same' - Convolutional Layer 5: - Filters: 32 - Kernel Size: (3, 3) - Activation: ReLU - Padding: 'same' 8. Max Pooling Layer 3: - Pool Size: (2, 2) - Strides: 1 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 128 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 20 (for 20 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf16	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (7, 7)	1. Dropout Layer 1: - Rate: 0.5	SGD momentum 0.9	0.3023	0.2944

- Activation: ReLU	- Seed: 42		
- Input Shape: (28, 28, 1)			
	2. Dropout		
2. Batch Normalization	Layer 2:		
Layer 1:	- Rate: 0.5		
- Applies batch	- Seed: 42		
normalization after			
Convolutional Layer 1.			
, i			
3. Max Pooling Layer 1:			
- Pool Size: (3, 3)			
- Strides: 1			
4. Convolutional Layer 2:			
- Filters: 64			
- Kernel Size: (2, 2)			
- Activation: ReLU			
- Padding: 'same'			
T adding: barrie			
5. Batch Normalization			
Layer 2:			
- Applies batch			
normalization after			
Convolutional Layer 2.			
Convolutional Layer 2.			
6. Max Pooling Layer 2:			
- Pool Size: (3, 3)			
- Fooi Size. (3, 3) - Strides: 2			
- Strides. 2			
7 Convolutional Layers 2 4			
7. Convolutional Layers 3, 4, 5:			
- Convolutional Layer 3: - Filters: 64			
- Kernel Size: (5, 5)			
- Activation: ReLU			
- Padding: 'same'			
- Convolutional Layer 4:			
- Filters: 64			
- Kernel Size: (4, 4)			
- Activation: ReLU			
- Padding: 'same'			
- Convolutional Layer 5:			

Conf17	1. Convolutional Layer 1: - Filters: 64 - Kernel Size: (7, 7) - Activation: ReLU - Input Shape: (28, 28, 1)	1. Dropout Layer 1: - Rate: 0.5 - Seed: 42 2. Dropout	momentum 0.9	10.2623	.∠549
	- Filters: 32 - Kernel Size: (10, 10) - Activation: ReLU - Padding: 'same' 8. Max Pooling Layer 3: - Pool Size: (3, 3) - Strides: 1 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 32 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 15 (for 15 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform 1. Convolutional Layer 1:	1. Dropout	SGD	0.2623	0.2549

2. Batch Norm	alization	Layer 2:		
Layer 1:		- Rate: 0.5		
- Applies batcl	n	- Seed: 42		
normalization	after			
Convolutional	Layer 1.			
3. Max Pooling	g Layer 1:			
- Pool Size: (3,	3)			
- Strides: 1				
4. Convolution	ıal Layer 2:			
- Filters: 64	-			
- Kernel Size: (8, 8)			
- Activation: R				
- Padding: 'sar				
5. Batch Norm	alization			
Layer 2:				
- Applies batcl	n			
normalization				
Convolutional				
6. Max Pooling	n Laver 2:			
- Pool Size: (3,				
- Strides: 2				
0.11466.2				
7. Convolution	al Layers 3, 4,			
5:	,, .,			
- Convolutiona	ıl Laver 3:			
- Filters: 64	- , ,			
- Kernel Size: (5, 5)			
- Activation: R	· · ·			
- Padding: 'sar				
- Convolutiona				
- Filters: 64	ii Eujoi 4.			
- Kernel Size: ((₄₋₄)			
- Activation: R	· · · ·			
- Padding: 'sar				
- Convolutiona				
- Filters: 64	ıı Layeı J.			
	10 10)			
- Kernel Size: (
- Activation: R	ELU			

	- Padding: 'same'				
	r adding. Junic				
	8. Max Pooling Layer 3:				
	- Pool Size: (3, 3)				
	- Strides: 1				
	9. Flatten Layer:				
	- Flattens the input to				
	prepare for the fully				
	connected layers.				
	10. Dense Layer 1:				
	- Units: 128				
	- Activation: ReLU				
	- Weight Initialization:				
	GlorotUniform				
	- Batch Normalization				
	11. Dense Layer 2:				
	- Units: 64				
	- Activation: ReLU				
	- Weight Initialization:				
	GlorotUniform				
	- Batch Normalization				
	12. Output Dense Layer:				
	- Units: 15 (for 15 classes in				
	classification)				
	- Activation: Softmax				
	- Weight Initialization:				
	GlorotUniform				
	1. Convolutional Layer 1:	1. Dropout	SGD	0.1505	0.2499
Conf18	- Filters: 64	Layer 1:	momentum		– 100
COMMITO	- Kernel Size: (6, 6)	- Rate: 0.05	0.9		
	- Activation: ReLU	- Seed: 42			
	- Input Shape: (28, 28, 1)				
		2. Dropout			
	2. Batch Normalization	Layer 2:			
	Layer 1:	- Rate: 0.05			
	- Applies batch	- Seed: 42			

normalization after		
Convolutional Layer 1.		
1		
3. Max Pooling Layer 1:		
1		
- Pool Size: (3, 3)		
- Strides: 1		
4. Convolutional Layer 2:		
- Filters: 64		
- Kernel Size: (5, 5)		
- Activation: ReLU		
- Padding: 'same'		
radulily. Sairie		
E Datah Nawasalisatian		
5. Batch Normalization		
Layer 2:		
- Applies batch		
normalization after		
Convolutional Layer 2.		
6. Max Pooling Layer 2:		
- Pool Size: (3, 3)		
- Strides: 1		
Strides. 1		
7 Compositional Layers 2 4		
7. Convolutional Layers 3, 4,		
5:		
- Convolutional Layer 3:		
- Filters: 64		
- Kernel Size: (4, 4)		
- Activation: ReLU		
- Padding: 'same'		
- Convolutional Layer 4:		
- Filters: 64		
- Kernel Size: (3, 3)		
- Activation: ReLU		
- Padding: 'same'		
- Convolutional Layer 5:		
- Filters: 64		
- Kernel Size: (2, 2)		
- Activation: ReLU		
- Padding: 'same'		
8. Max Pooling Layer 3:		

	- Strides: 1 9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 15 (for 15 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform	1 Dropout	SCD	0.1476	0.2496
Conf19	1. Convolutional Layer 1: - Filters: 64 - Kernel Size: (8, 8) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1.	1. Dropout Layer 1: - Rate: 0.15 - Seed: 42 2. Dropout Layer 2: - Rate: 0.15 - Seed: 42	SGD momentum 0.9	0.1476	0.2496

3	3. Max Pooling Layer 1:
	Pool Size: (3, 3)
	Strides: 2
	Strides. 2
	. Convolutional Layer 2:
-	Filters: 64
_	Kernel Size: (5, 5)
	Activation: ReLU
	Padding: 'same'
	r ddding. Sume
,	Details Norman Reserving
	5. Batch Normalization
	ayer 2:
-	Applies batch
r	ormalization after
	Convolutional Layer 2.
	o. Max Pooling Layer 2:
	Pool Size: (3, 3)
	Strides: 2
	'. Convolutional Layers 3, 4,
	j:
-	Convolutional Layer 3:
_	Filters: 64
-	Kernel Size: (4, 4)
	Activation: ReLU
	Padding: 'same'
	Convolutional Layer 4:
	Filters: 64
-	Kernel Size: (4, 4)
-	Activation: ReLU
-	Padding: 'same'
-	Convolutional Layer 5:
_	Filters: 64
	Kernel Size: (4, 4)
I I	Activation: ReLU
	Padding: 'same'
	B. Max Pooling Layer 3:
-	Pool Size: (3, 3)
-	Strides: 2

	9. Flatten Layer: - Flattens the input to prepare for the fully connected layers. 10. Dense Layer 1: - Units: 128 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 11. Dense Layer 2: - Units: 64 - Activation: ReLU - Weight Initialization: GlorotUniform - Batch Normalization 12. Output Dense Layer: - Units: 25 (for 25 classes in classification) - Activation: Softmax - Weight Initialization: GlorotUniform				
Conf20	1. Convolutional Layer 1: - Filters: 32 - Kernel Size: (6, 6) - Activation: ReLU - Input Shape: (28, 28, 1) 2. Batch Normalization Layer 1: - Applies batch normalization after Convolutional Layer 1. 3. Max Pooling Layer 1: - Pool Size: (3, 3) - Strides: 1	1. Dropout Layer 1: - Rate: 0.25 - Seed: 42 2. Dropout Layer 2: - Rate: 0.25 - Seed: 42	SGD momentum 0.9	0.2094	0.2404

_	,	 	
	4. Convolutional Layer 2:		
	- Filters: 64		
	- Kernel Size: (6, 6)		
	- Activation: ReLU		
	- Padding: 'same'		
	5. Batch Normalization		
	Layer 2:		
	- Applies batch		
	normalization after		
	Convolutional Layer 2.		
	6. Max Pooling Layer 2:		
	- Pool Size: (3, 3)		
	- Strides: 2		
	7. Convolutional Layers 3, 4,		
	5:		
	- Convolutional Layer 3:		
	- Filters: 64		
	- Kernel Size: (4, 4)		
	- Activation: ReLU		
	- Padding: 'same'		
	- Convolutional Layer 4:		
	- Filters: 64		
	- Kernel Size: (4, 4)		
	- Activation: ReLU		
	- Padding: 'same'		
	- Convolutional Layer 5:		
	- Filters: 32		
	- Kernel Size: (4, 4)		
	- Activation: ReLU		
	- Padding: 'same'		
	8. Max Pooling Layer 3:		
	- Pool Size: (3, 3)		
	- Strides: 1		
	9. Flatten Layer:		
	- Flattens the input to		
	prepare for the fully		
	1		

connected layers.			
10. Dense Layer 1: - Units: 64 - Activation: ReLU - Weight Initializatio GlorotUniform - Batch Normalizatio			
11. Dense Layer 2: - Units: 32 - Activation: ReLU - Weight Initializatio GlorotUniform - Batch Normalizatio			
12. Output Dense La - Units: 10 (for 10 cl classification) - Activation: Softma - Weight Initializatio GlorotUniform	asses in		

Analisis

Pendekatan menggunakan arsitektur yang lebih kompleks seperti AlexNet pada dataset Fashion-MNIST yang relatif sederhana menghasilkan model yang kurang bagus, karena dataset Fashion-MNIST adalah gambar hitam-putih dan ukurannya lebih kecil, tidak seperti CIFAR yang memiliki gambar berwarna dan ukuran lebih besar.

Awalnya kami mencoba menggunakan hyperparameter default dari AlexNet kami mendapatkan accuracy yang lumayan bagus namun loss nya masih tinggi / tidak stabil. Kemudian kami mencoba beberapa konfigurasi dan menemukan bahwa semakin simple modelnya semakin baik accuracy dan lossnya.

Jadi, dalam konteks Fashion-MNIST penggunaan model yang lebih sederhana atau arsitektur yang lebih ringan seringkali lebih cocok dan dapat memberikan kinerja yang baik tanpa mengorbankan waktu training atau computational cost yang tinggi.

Reference

Convolutional Neural Network (CNN) | TensorFlow Core Conv2D layer

□ Image classification using CNN (CIFAR10 dataset) | Deep Learning Tutorial 24 (Tensorflo...