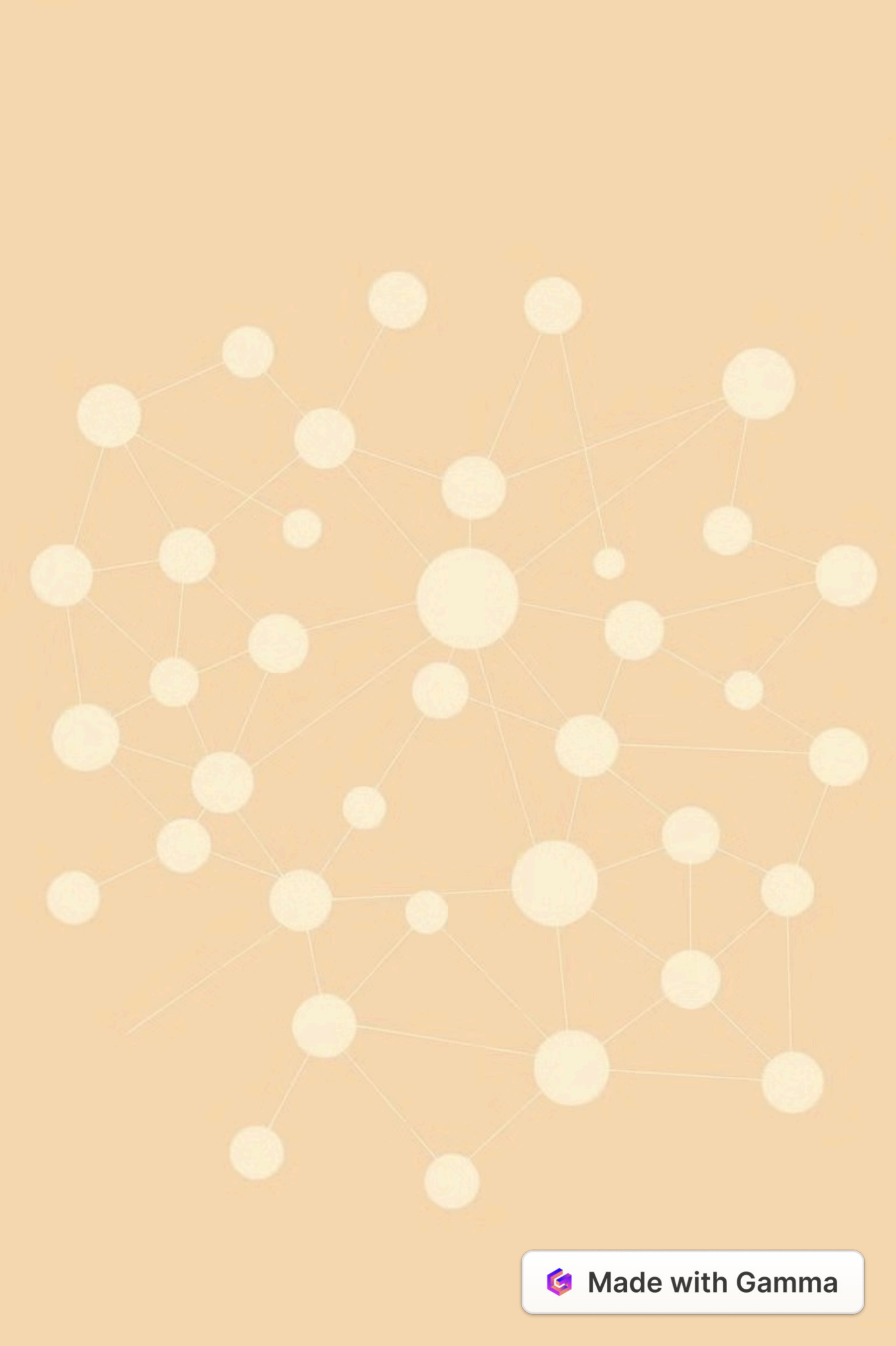
Comparative Analysis of ANN and CNN Models

Exploring the performance of Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) for regression and classification tasks using PyTorch and Keras.

S by Sadia Shakoor



Datasets

California Housing

Regression, 20,640 records, 8 features, median house value target.

Customer Churn

Classification, 7,043 records, 20 features, binary churn target.

CIFAR-10

Classification, 60,000 images, 32x32 RGB, 10 classes.

PyTorch Model Architectures

1
PyTorch ANN (Regression)

32-16-8-1 layers, ReLU activation.

2
PyTorch ANN (Classification)

16-8-1 layers, ReLU activation, sigmoid output.

CNN Model Architectures

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 8)	224
max_pooling2d (MaxPooling2D)	(None, 16, 16, 8)	0
conv2d_1 (Conv2D)	(None, 16, 16, 16)	1,168
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 16)	0
conv2d_2 (Conv2D)	(None, 8, 8, 32)	4,640
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 32)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 128)	65,664
dense_1 (Dense)	(None, 10)	1,290

Total params: 72,986 (285.10 KB)
Trainable params: 72,986 (285.10 KB)
Non-trainable params: 0 (0.00 B)

Model 1: Baseline CNN

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_4 (Conv2D)	(None, 16, 16, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_5 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_2 (Dense)	(None, 128)	262,272
dense_3 (Dense)	(None, 10)	1,290

Total params: 356,810 (1.36 MB)
Trainable params: 356,810 (1.36 MB)
Non-trainable params: 0 (0.00 B)

Model 2: Increased Filter Size

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d_12 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_13 (Conv2D)	(None, 16, 16, 64)	18,496
max_pooling2d_13 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_14 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling2d_14 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_4 (Flatten)	(None, 2048)	0
dense_10 (Dense)	(None, 128)	262,272
dense_11 (Dense)	(None, 64)	8,256
dense_12 (Dense)	(None, 10)	650

Total params: 364,426 (1.39 MB)
Trainable params: 364,426 (1.39 MB)
Non-trainable params: 0 (0.00 B)

Model 3: Additional Dense Layer

Training Configurations

Regression

SGD optimizer, learning rates 0.01, 0.1, epochs 30, 50, batch sizes len(X_train)/10, len(X_train)/5 and len(X_train).

Classification

Adam optimizer, learning rates 0.01, 0.1, epochs 30, 100, batch sizes len(X_train)/10, len(X_train)/5 and len(X_train).

CNN

Adam optimizer, learning rate 0.001, epochs 10, 15, 20, batch size 32 and 64.

Performance Metrics

1.31

MSE

Regression, epoch 30, learning rate 0.01.

83%

Accuracy

Classification, epoch 100, learning rate 0.01.

0.90

MAE

Regression, epoch 30, learning rate 0.01.

74%

Accuracy

CNN Model 2, epoch 20, learning rate 0.001.

Visualizations

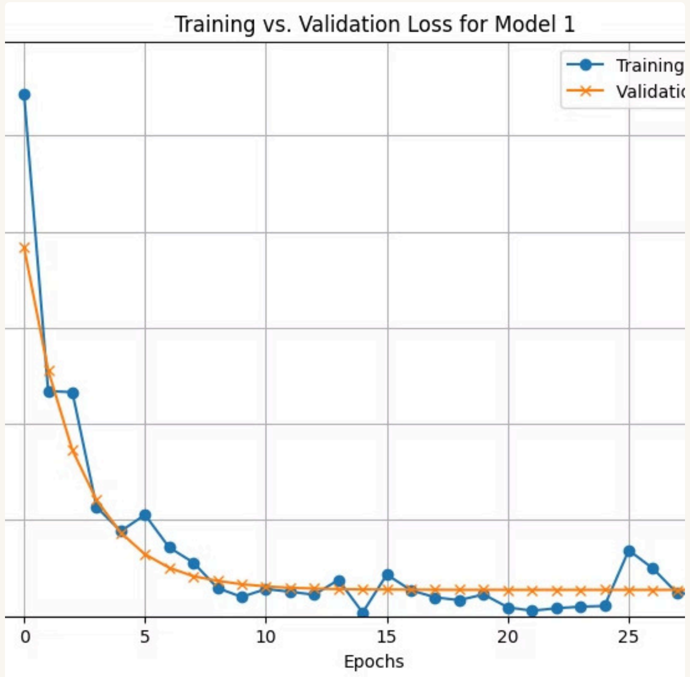
Learning Curves

Training and validation loss, accuracy over epochs.

Confusion Matrices

Class-wise performance for classification tasks.

Learning Curves for Regression Models



Model 1

Optimizer =SGD

Learning Rate =0.01

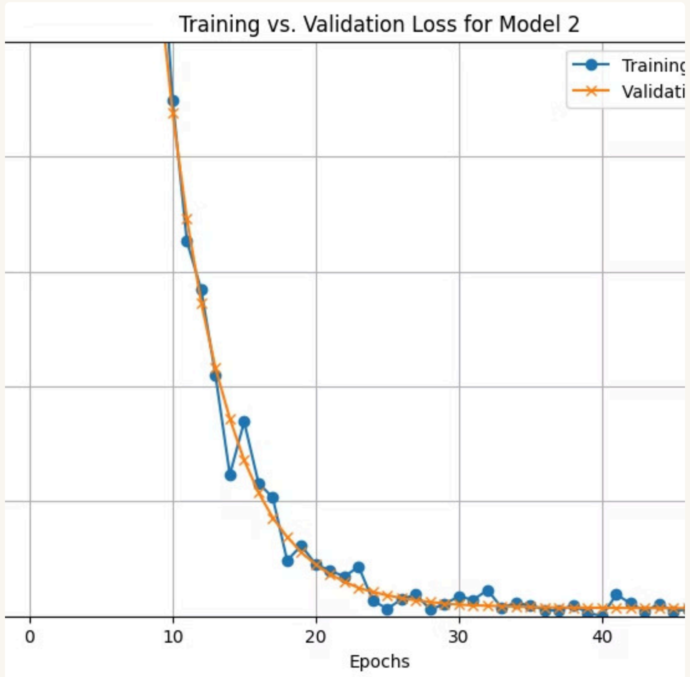
Loss Fun = MSELoss

Epoch=30

Batch Size=int(len(X_train)/10)

Mean Squared Error: 1.3105

Mean Absolute Error:0.90097



Model 2

Optimizer =SGD

Learning Rate =0.01

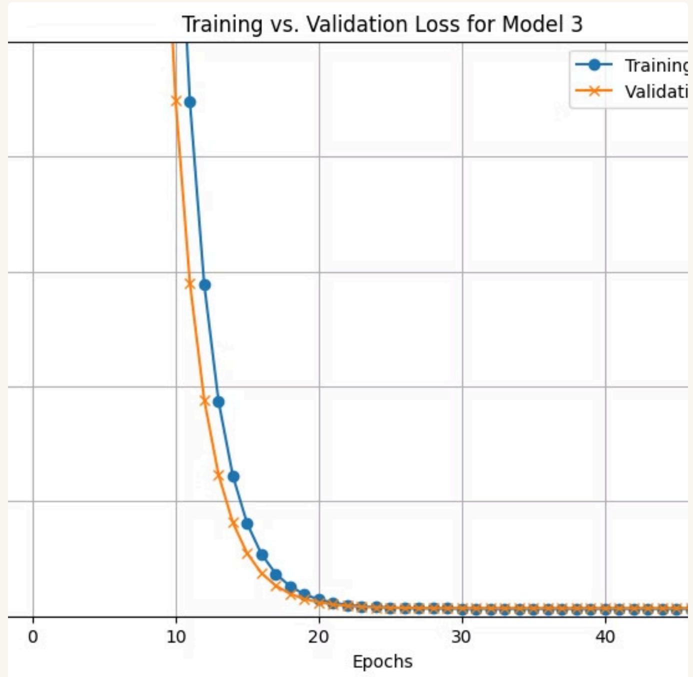
Loss Fun = MSELoss

Epoch=100

Batch Size=int(len(X_train)/5)

Mean Squared Error: 1.3218

Mean Absolute Error: 0.9261



Model 3

Optimizer =SGD

Learning Rate =0.01

Loss Fun = MSELoss

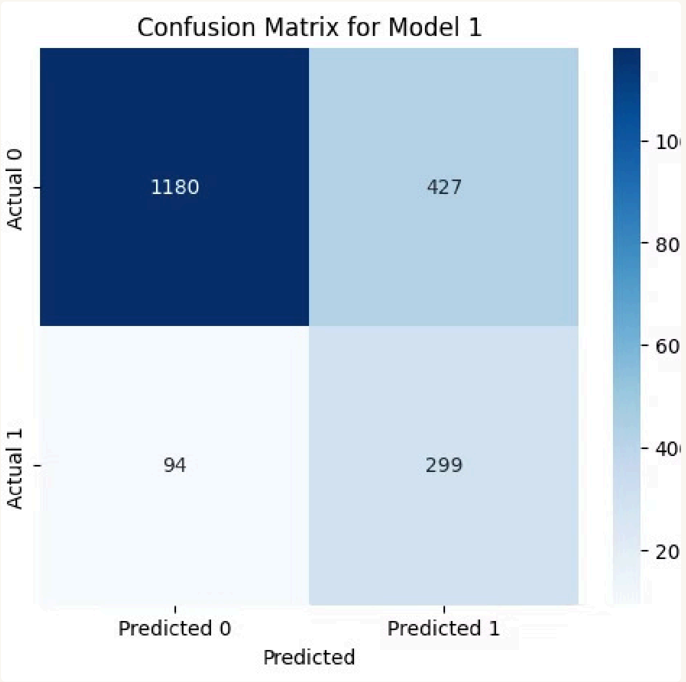
Epoch=100

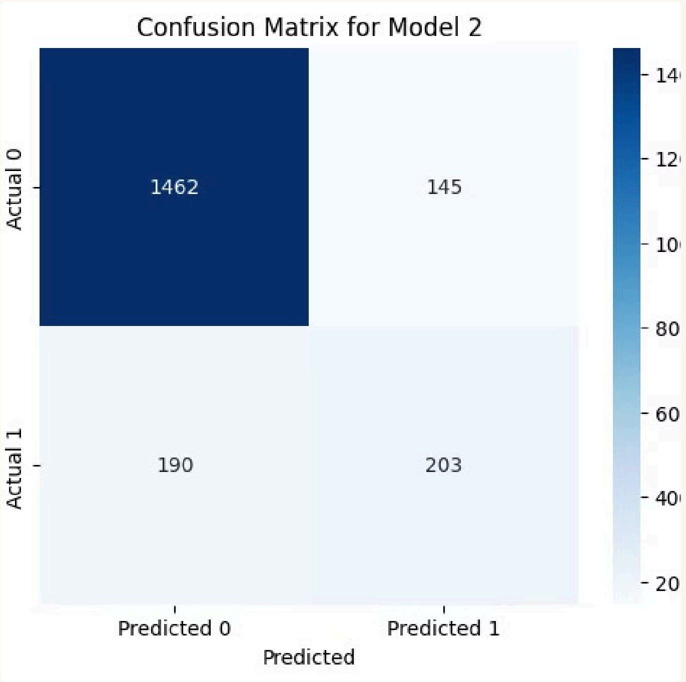
Batch Size=int(len(X_train))

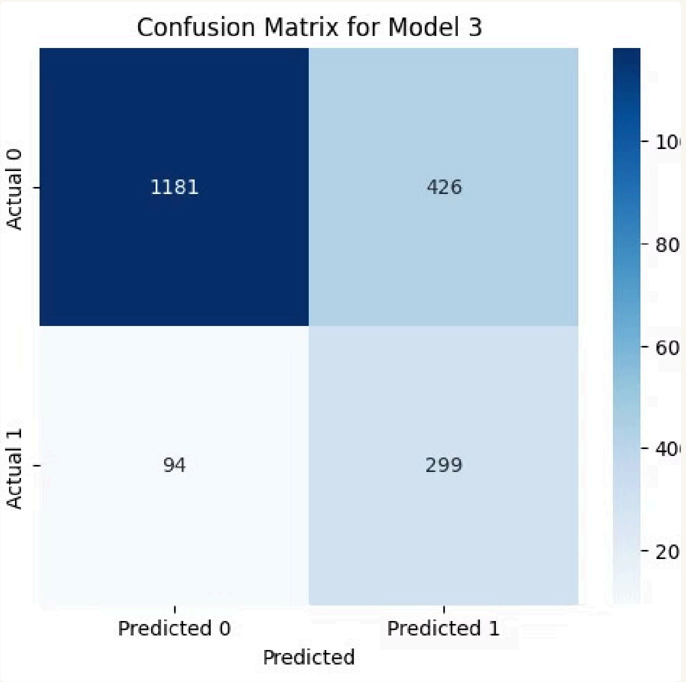
Mean Squared Error: 1.3218

Mean Absolute Error: 0.92607

Confusion Matrices for Classification Model







Model 1

Optimizer = Adam

Learning Rate =0.01

Loss Fun = BCELoss

Epoch= 30

Batch Size=int(len(X_train)/10)

Accuracy: 0.7395

Model 2

Optimizer = Adam

Learning Rate =0.01

Loss Fun = BCELoss

Epoch= 100

Batch Size=int(len(X_train)/5)

Accuracy: 0.8325

Model 3

Optimizer = Adam

Learning Rate =0.1

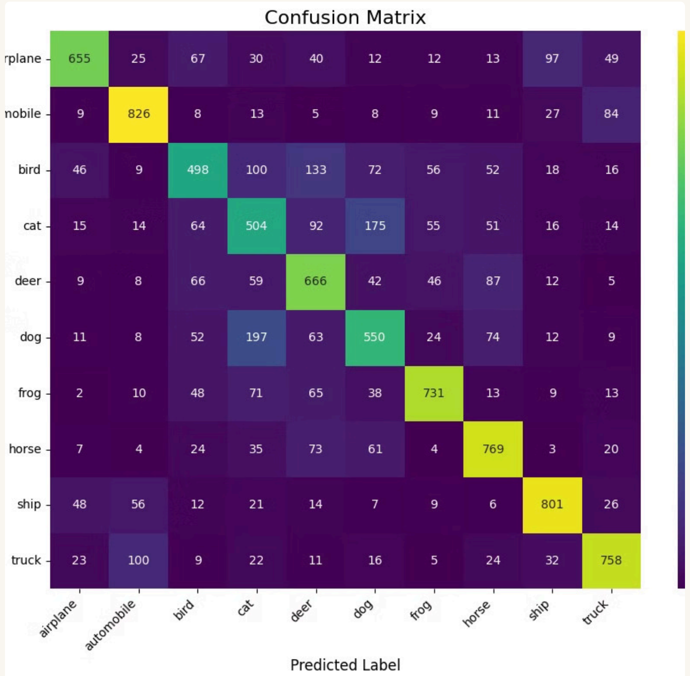
Loss Fun = BCELoss

Epoch= 100

Batch Size=int(len(X_train))

Accuracy: 0.74

Confusion Matrices for CNN Models



Model 1

Optimizer = Adam

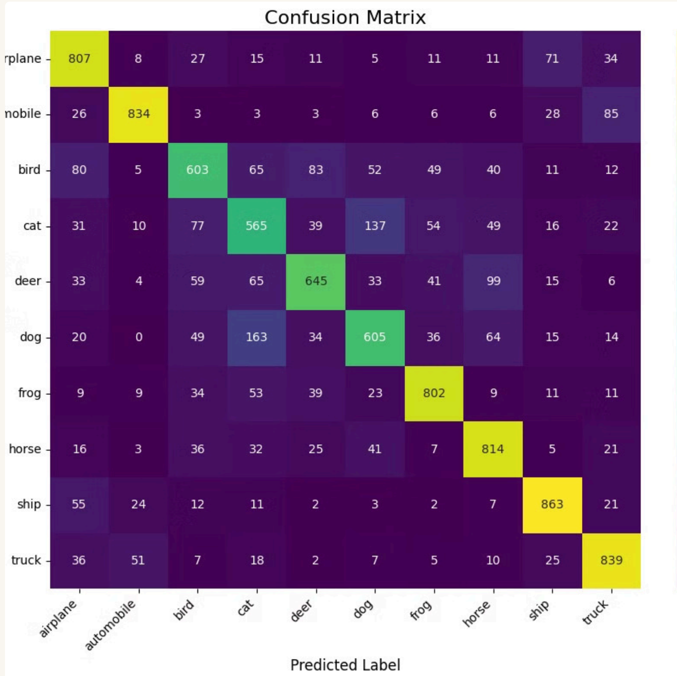
Learning Rate =0.01

Loss Fun = BCELoss

Epoch=10

Batch Size=32

Accuracy= 0.68 10000



Model 2

Optimizer = Adam

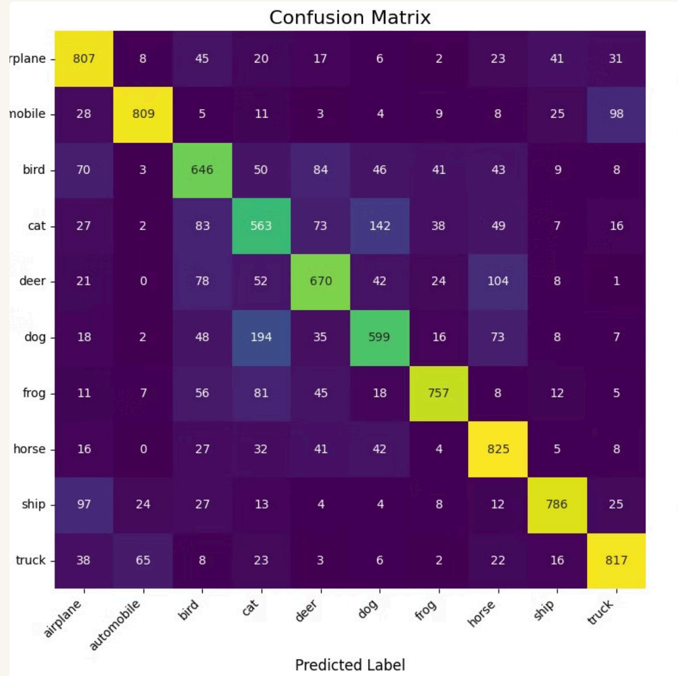
Learning Rate =0.01

Loss Fun = BCELoss

Epoch= 20

Batch Size= 64

Accuracy= 0.74



Model 3

Optimizer = Adam

Learning Rate =0.001

Loss Fun = BCELoss

Epoch= 15

Batch Size= 32

Accuracy= 0.73

Comparative Table

Model	Dataset / Task	Key Hyperparams	Final Metric	Training Time
PyTorch ANN (Reg)	California Housing	LR=0.01, Epoch=30	MSE=1.31, MAE=0.90	~1 min
PyTorch ANN (Class)	Customer Churn	LR=0.01, Epoch=100	Accuracy=83%, Recall=52%	~1 min
Keras CNN (Model 2)	CIFAR-10	LR=0.001, Epoch=20	Accuracy=74%	~1 min (GPU)

Key Takeaways

1

Strengths

PyTorch ANN: Simple architectures, decent results for regression and binary classification.

2

Strengths

Keras CNN: Improved accuracy with deeper architectures and dropout regularization.

3

Weaknesses

PyTorch ANN: Limited recall for imbalanced datasets.

4

Weaknesses

Keras CNN: Performance plateaued with increased complexity.