CIFAR10

Gwen, Xindi, Olivia, Malka, Salena, Joyce

Table of contents

Table of contents

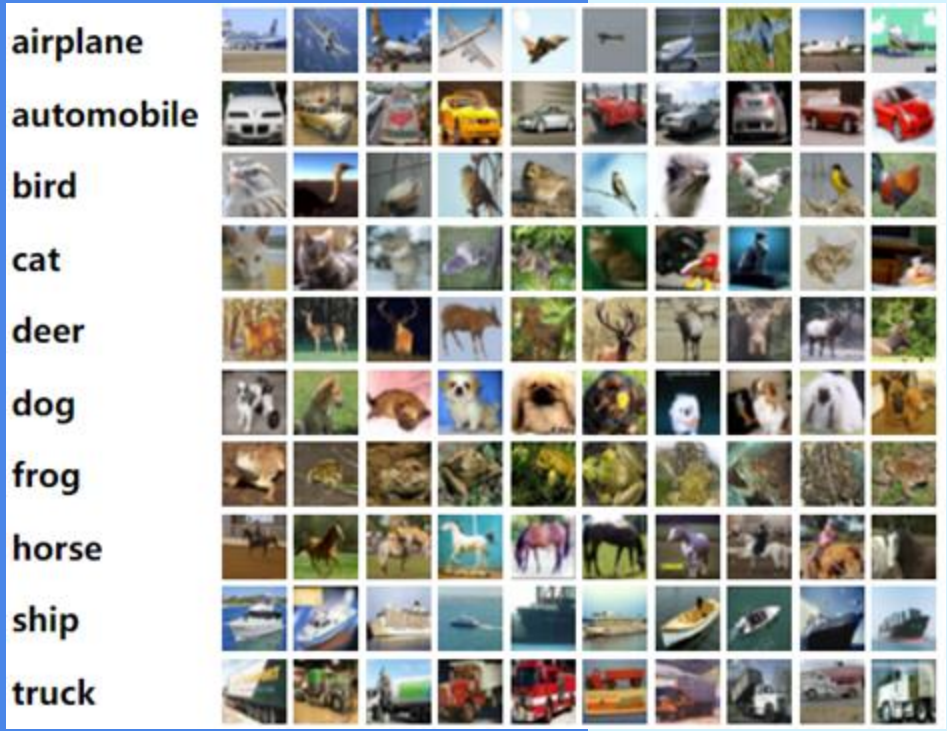
Introduction

CIFAR10:

- 10 different categories
- 60,000 images

Task:

- Create and train a neural network (CNN) for calibrated for accuracy in image classification
    - Image classification:
        - method to classify images into specific categories



ResNet Architecture

Structure:

- Uses ResNet 50 V2
- Multiple dense layers with normalization and dropouts after each one

Theory:

- Multiple convolutional layers
- Skip connections
  - Prevents explosions and vanishing gradients
- Normalization and dropouts to prevent overfitti

successful)

```
Total params:
26,159,370
```

```
base_model = applications.ResNet50V2(
    weights='imagenet',  # Load weights pre-trained on the ImageNet dataset
    input_shape=(224, 224, 3),
    include_top=False) # do not include the classifier at the top
```

```
#build layers
model = models.Sequential([
    layers.Resizing(224, 224),
    layers.RandomFlip("horizontal_and_vertical"),
    base_model,
    layers.Conv2D(128, (3, 3), activation = "relu"),
    layers.GlobalAveragePooling2D(),

    layers.Dense(512),
    layers.BatchNormalization(),
    layers.Dropout(0.1),
    layers.Activation("relu"),
    layers.Dense(256),
    layers.BatchNormalization(),
    layers.Dropout(0.1),
    layers.Activation("relu"),
    layers.Dense(128),
    layers.BatchNormalization(),
    layers.Dropout(0.1),
    layers.Activation("relu"),

    layers.Dense(10, activation = "softmax"),
])

model.layers[2].trainable = False

model.build((None, 32, 32, 3))
```

Network Architecture

Structure:

- Operates in blocks
- Each would convolute, normalize, activate, and use dropout
- Was flattened then put through dense layers

Theory:

- The neurons fan out for convolute and in for dense to preserve the most data possible
- Dropouts after every block to prevent overfitting (partially successful)

```
#build layers
model = models.Sequential([
    Conv2D(32, (3, 3), input_shape=(32, 32, 3)),
    BatchNormalization(),
    Activation('relu'),
    Dropout(.05),

    Conv2D(64, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    Dropout(.1),

    Conv2D(128, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    Dropout(.1),

    Conv2D(128, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    Dropout(.1),

    Conv2D(256, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    Dropout(.1),

    Conv2D(256, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    Dropout(.1),

    Conv2D(512, (3, 3)),
    BatchNormalization(),
    Activation('relu'),

    Flatten(),
    Dense(512),
    BatchNormalization(),
    Activation('relu'),
    Dense(128),
    BatchNormalization(),
    Activation('relu'),
    Dense(10, activation='softmax')
])
```

```
Total params:
87,316,426
```

Hyperparameters

Optimizer → Adam

Batch_size

ResNet: 128 Fan: 64

Epochs

ResNet: 3 Fan: 30

Workers

Activation → ReLU, Softmax

Model Performance

ResNet

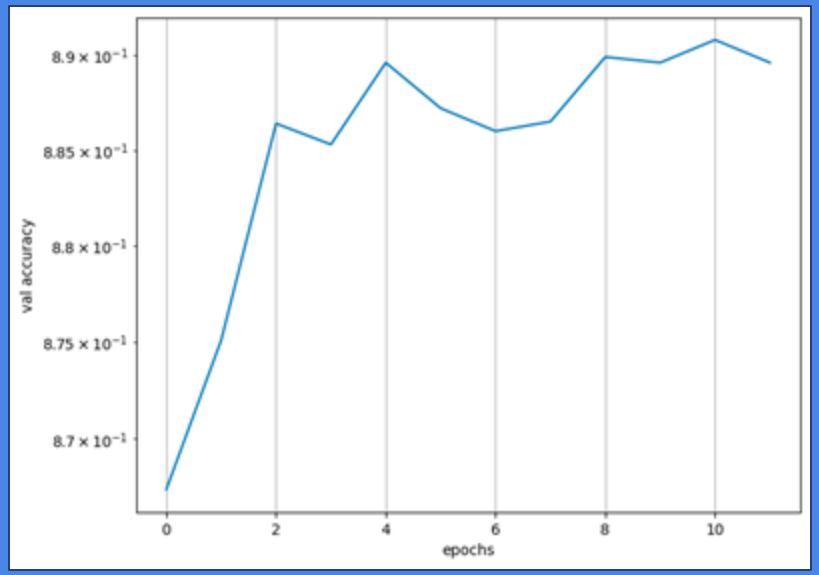
Train acc: 95.59%
Val acc: 89.22%
Test acc: 84.51%

Fan

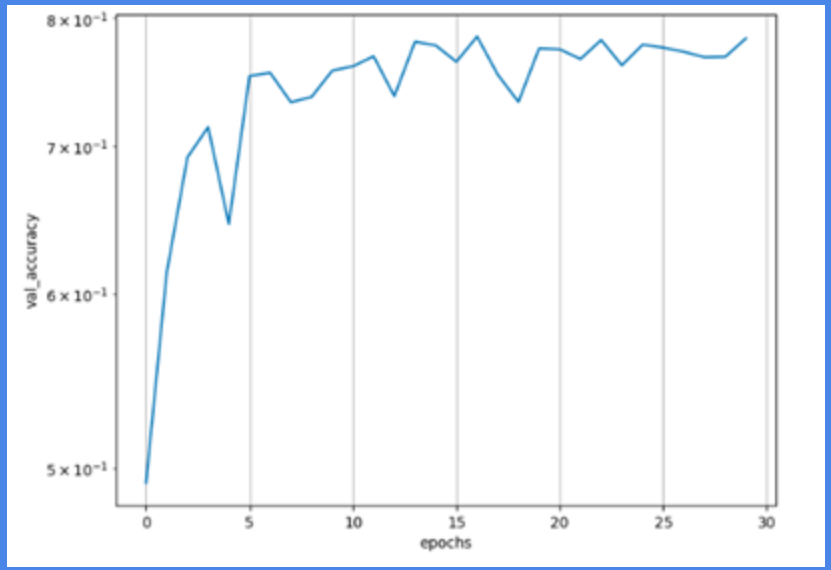
Train acc: 97.79%
Val acc: 78.04%
Test acc: 76.00%

Progression of Val Accuracy

ResNet



Fan Model



Challenges

Problems

- Epoch run time errors
- Overfitting
- Resnet adaptation
- 10% val_accuracy

Solutions

- GP4 and include_top=False
- Regularization
- Checking runtime type
- Fanning in correct direction

Conclusion

ResNet > Fan