Computer Vision

Ruba Alsulami -2110618

Content

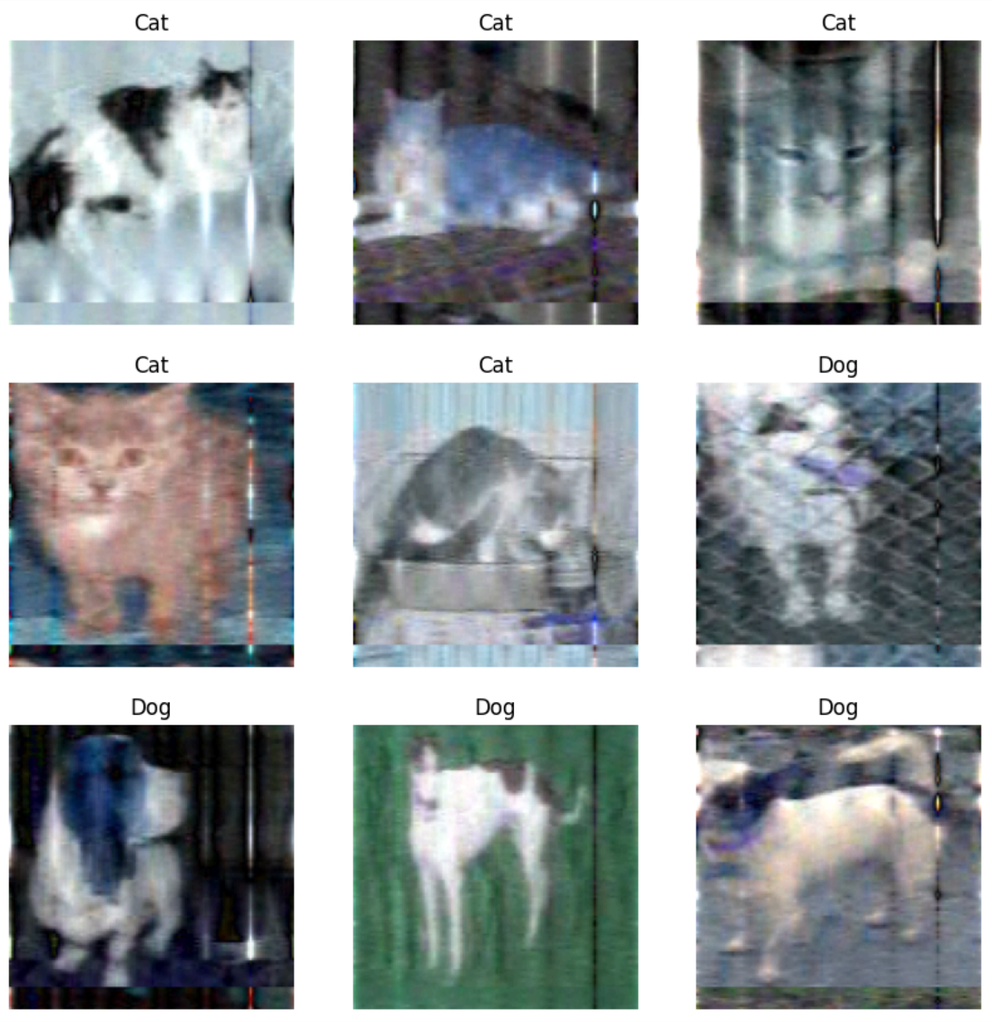
1. **Preprocessing on motion blur images**
2. **Preprocessing on salt-pepper noise images**
3. **Augmentation**
4. **Model & Training**
5. **Results**

Motion Blur Preprocessing

For motion blur we used implements Wiener filtering to reduce motion blur in color images by applying a motion blur PSF and processing each color channel separately. This method leverages frequency domain techniques for improved image restoration.

Two parameters are defined:

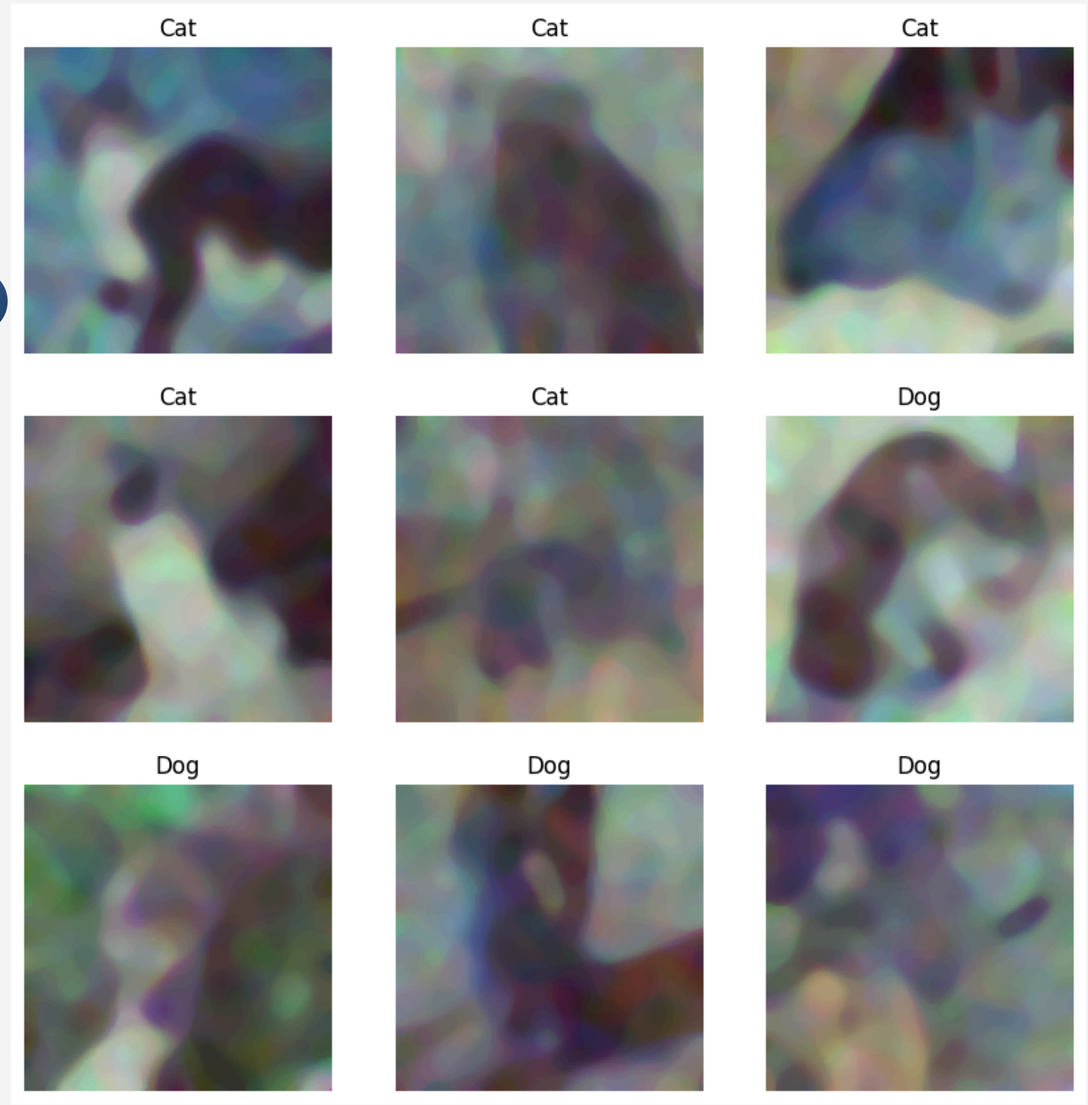
- The size of the kernel (20x20 pixels) which determines the extent of the blur.
- The angle of the motion blur (0 degrees in this case, indicating horizontal motion).



Salt-Pepper Noise Preprocessing

We used median filter to process images affected by salt-and-pepper noise.

- We set the kernel size to 9, meaning that for each pixel, the filter considers a 9x9 window of neighboring pixels.
- The filtering process is applied iteratively 4 times. Each iteration refines the result further, allowing the filter to more effectively eliminate noise.



Augmentation

```
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Model & Training

Model Architecture

Consists of 3 Blocks, each of them contains:

- 2 Conv2D layer
- Batch Normalization Layer
- Maxpooling Layer
- Dropout (probability of 0.3)

For the classification head:

- 2 Dense layers with 128 hidden units
- Dropout layers (probability of 0.5) after each dense layer

Training Process

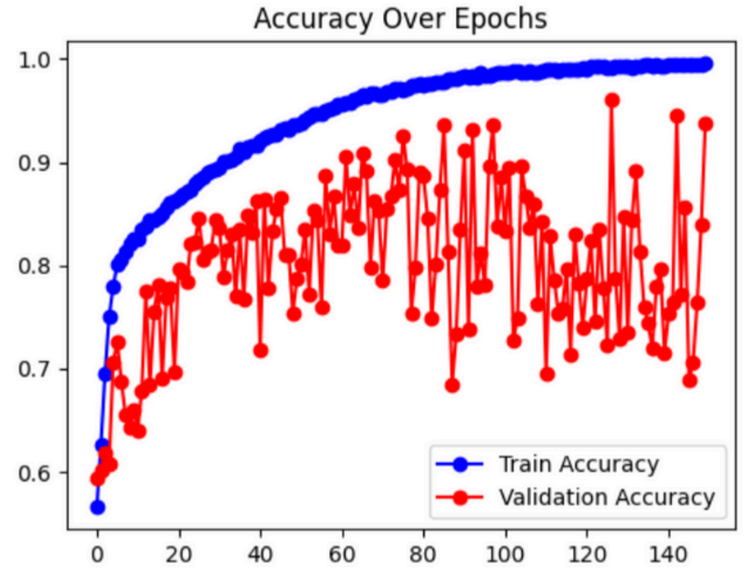
- **L**earning rate 0.00001.
- Trained on 150 epoch

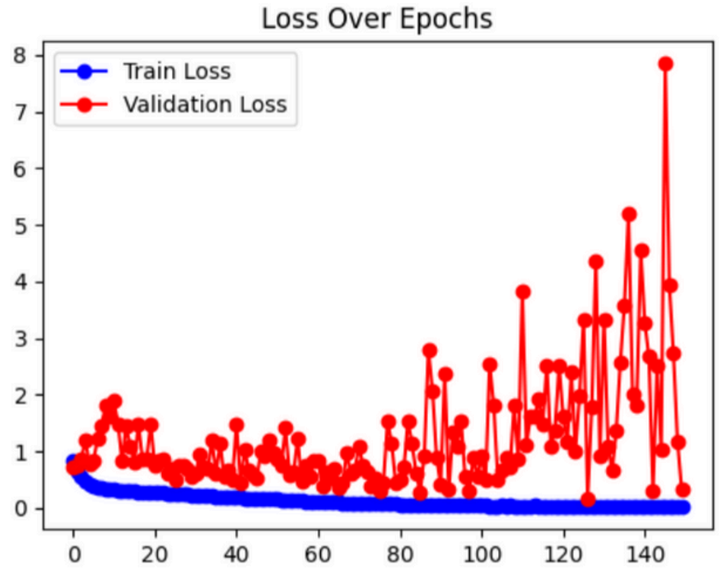
```
# Define the CNN model
model = Sequential([
    # First block
    Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(img_size, img_size, 3)),
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    # Second block
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    # Third block
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    # Fully connected layers
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  # Binary classification
])
```

Results

Training

```
Epoch 150/150
500/500 ━━━━━━━━━━━━━━━━━━━━ 23s 46ms/step - accuracy: 0.9970 - loss: 0.0105 - val_accuracy: 0.9380 - val_loss: 0.3363
```



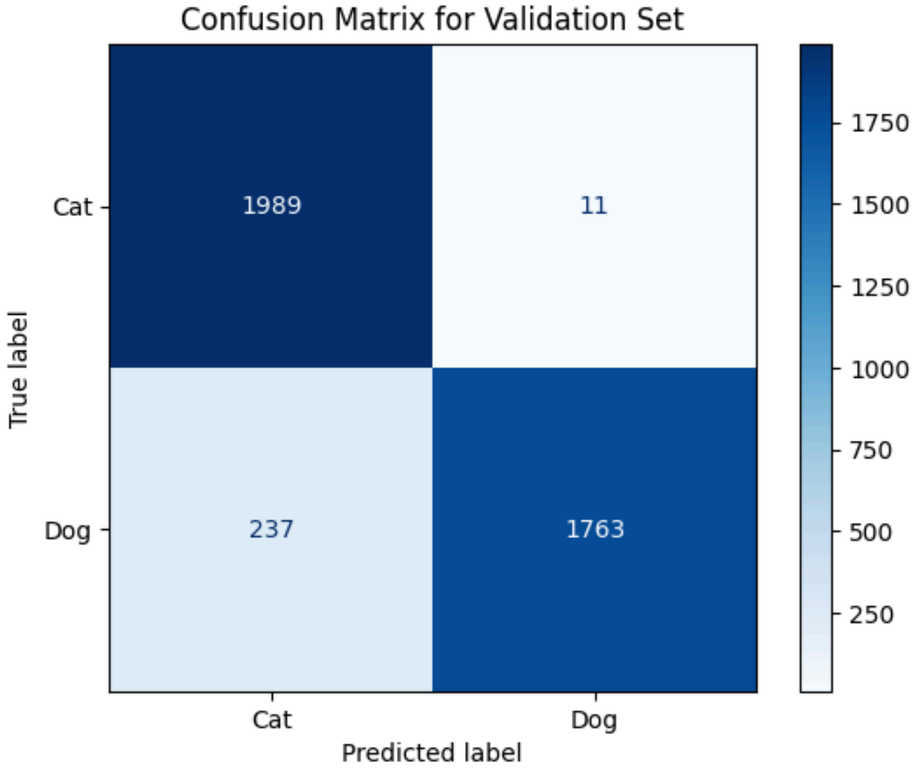


Validation

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.9830 - loss: 0.0834
Accuracy: 0.9380
Loss: 0.3363
```

Results


True: dog
Pred: dog
True: dog
Pred: dog
True: cat
Pred: cat
True: dog
Pred: dog
True: cat
Pred: cat
True: dog
Pred: dog
True: dog
Pred: dog
True: cat
Pred: cat
True: cat
Pred: cat
True: cat
Pred: cat


Confusion Matrix for Validation Set
Cat
Dog
True label
1989
11
237
1763
Cat
Dog
Predicted label
1750
1500
1250
1000
750
500
250

Thanks