Classification of 9 species of fish using a CNN

Black Sea Sprat, Gilt-Head Bream, Horse Mackerel, Red Mullet, Red Sea Bream, Sea Bass, Shrimp, Striped Red Mullet, Trout

Understanding the data

The dataset is comprised of 9 different species of fish. Each species of fish contains 2 folders, one where the fish image is full RGB and the other block black and white seen in figure 1 and 2.

There are 2 individual fish used for each set of images for each fish. The images of the fish are rotated around 360°. New photos of fish tested on the trained network may be taken with different angles of rotation, thus it is essential that the network is trained with data that displays all angles.

The fish on their own don't possess much complexity with the most identifiable factors being, shape and colour. Right away we can guess that training the network with the RGB dataset will provide a greater accuracy. Full RGB images contain 3 times as much data as their greyscale counterpart.

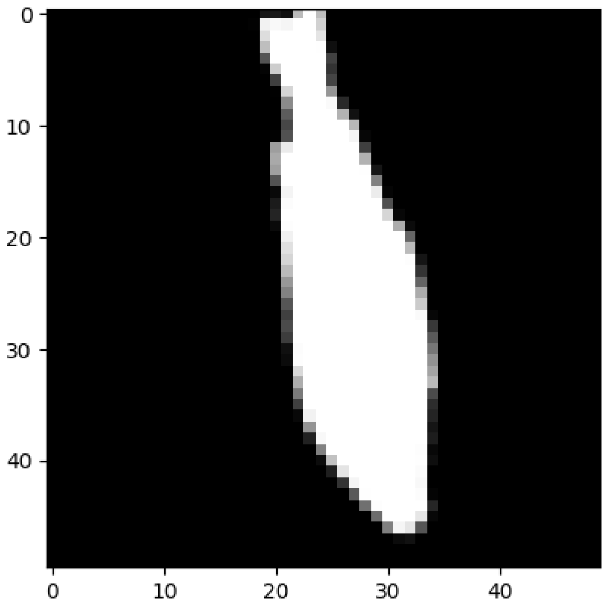

Figure 1 - 50x50 pixel image of fish in block black and white

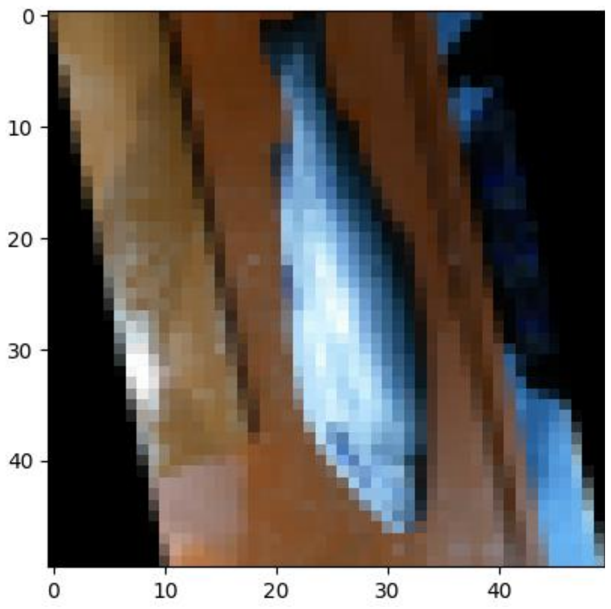

Figure 2 - 50x50 pixel image of fish in RGB

Importing, sorting, and compressing the data

The images have dimensions 590x445, this is far too large for the CNN to train on in a reasonable time. Excluding the time the accuracy of the model would not improve much if I were to train the model with the source image dimensions. I will test the images from 10x10 up to 50x50. Making the images square has no impact on the accuracy.

I created a function that takes three parameters img_size, class_name and path_to_img. IMG_SIZE is the value of pixel in both the x and y axis.

Importing and compressing the images into python is trivial and done with cv2 functions. These images are useless without a label assigned. I chose 0-8 for each species of fish which is assigned with each image when imported done with class_num. This is so the CNN knows if its guess is correct.

```
class_num = class_name.index(category)
```

Once all the images have been read in python alongside their label, the lists of each are shuffled. This is to make sure the CNN doesn’t learn patterns that may occur from the ordered data.

```
def load_images_from_folder(img_size,class_name,path_to_img):
    training_data = []
    X = []
    y = []
    for category in class_name:
        path = os.path.join(path_to_img,category)
        class_num = class_name.index(category)
        for img in os.listdir(path):
            img = cv2.imread(os.path.join(path, img))
            img = cv2.resize(img, (img_size,img_size), 
interpolation = cv2.INTER_AREA)
            training_data.append([img,class_num])
    for features,label in training_data:
        X.append(features)
        y.append(label)
    random.shuffle(training_data)
    return X,y
```

Designing the neural network

For image classification a CNN yields the greatest accuracy. The CNN applies filters (matrices of differing values) which take a given kernel_size, in my two Conv2d layers I use (3,3) and (1,1). Each filter is scanned across each pixel takes the dot product with corresponding kernel_size pixel matrix to create a new image. These filters detect different features such as vertical and horizontal lines which can be used to identify fish in this case. As the fish in the dataset are moderately basic in terms of the number of features the Conv2d layers do not have to be very expensive computationally.

Each Covn2d layer is followed by a MaxPooling2D layer. The MaxPooling2D layer decreases the image data by taking the maximum value in a given matrix size ((2,2)) and outputs this as a matrix. Then a Flatten layer converts the data to 1 dimension so the Dense layers can read the data. For the RGB case the matrix's become tensors.

```
model.add(Conv2D(32, (3,3),
input_shape=(img_size,img_size,3), activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (1,1),
input_shape=(img_size,img_size,3), activation='relu'))
model.add(MaxPooling2D(2,2))
```

The following layers then classify which fish the network has been fed. Two Dense layers with the activation function 'relu' followed by a Dense layer with 9 units which correspond to each fish that we want to be able to detect. The activation function 'softmax' is used for the final Dense layer in a classification neural network. This is because it normalises the outputs between 0 and 1 which corresponds to the probability of which fish it is. And whichever numbered neuron has the greatest probability is the guess as to which fish it has been given.

```
model.add(Dense(units=512, activation='relu'))
model.add(Dense(units=256, activation='relu'))
model.add(Dense(units=9, activation='softmax'))
```

The network is then complied, trained, and used to predict which fish has the highest probability.

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_num, verbose=1,
validation_data=(X_test,y_test))
val_loss, val_acc = model.evaluate(X_test,y_test)
```

Testing both sets of images

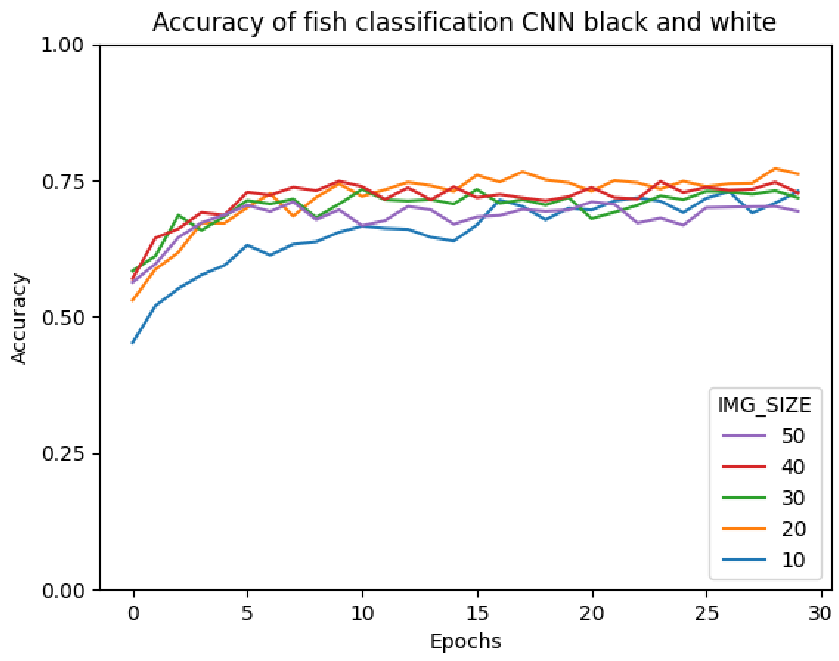

Figure 3

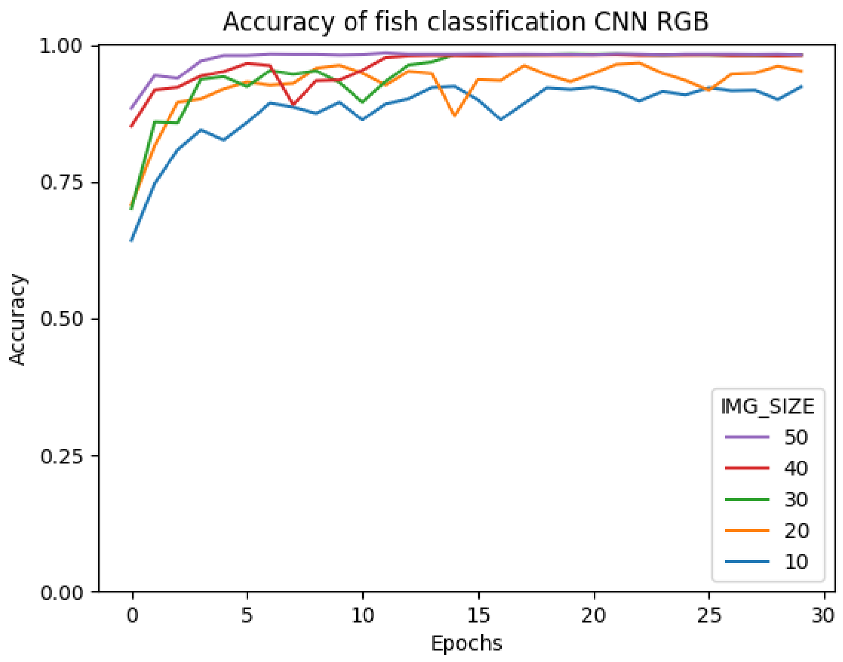

Figure 4

Figure 3 was trained on the block black and white dataset. The accuracy of the tested images never manages to converge for any pixel dimension. Whereas figure 4 displays the RGB trained dataset where an image dimension of 30x30 and above manage to converge at an accuracy of 98% and above. This is because the species differ in colour, not only shape and size.

When the black and white images are used to train the CNN, the accuracy never converges. The accuracy averages at 74%, with a peak accuracy of 76.2%. Although 76% seems high, when you are trying to identify an image, this cannot be used confidently making it useless. The image dimensions of the image play almost no role in how accurate the CNN can achieve. As seen in figure 3 the CNN ended up with the IMG_SIZE of 20 being the most accurate, but if I were to increase the number of epochs or run the CNN again it may train differently. None of the black and white trained CNNs managed to converge.

The accuracy of the RGB image trained CNN from IMG_SIZE 30 and above converges at ≈98.4%, with a peak of 98.7%. With such a high accuracy the CNN would be able to be used with confidence. For IMG_SIZE's 10 and 20 the convolutional layers are unable to distinguish which features belong to which fish species, thus the network accuracy never converges. Unlike the black and white trained CNN the image dimension has a significant impact on the accuracy to a certain point.

Dataset – Kaggle

A Large Scale Fish Dataset - Oğuzhan Ulucan