When looking at cost function val_loss and val_accuracy comprehensively, the best result I achieved was:
loss: 1.2612 - accuracy: 0.5559 - val_loss: 1.4089 - val_accuracy: 0.5243.
Below is a table of all the models I've tried and the best result I recorded.

Models	Result	epoch
model = tf.keras.Sequential([tf.keras.layers.Flatten(input_shape = (img_size, img_size, 3)), tf.keras.layers.Dense(128, activation='relu'), tf.keras.layers.Dropout(dropout), tf.keras.layers.Dense(num_classes, activation='softmax')])	loss: 2.3825 - accuracy: 0.1314 - val_loss: 2.3775 - val_accuracy: 0.1536	10

model = Sequential()
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(img_size, img_size, 3))) # Layer 1
model.add(layers.MaxPool2D(pool_size=(2,2))) # Layer 2
# Add more depth to CNN 	
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPool2D((2, 2)))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
# Add dense layers 	
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
model.add(layers.Dense(11, activation='softmax'))
	loss: 0.6870 - accuracy: 0.7740 - val_loss: 6.5362 - val_accuracy: 0.1782	10

vgg16_model = tf.keras.applications.vgg16.VGG16(include_top = False, weights = 'imagenet', pooling = 'maxpooling', input_shape =	loss: 1.2718 - accuracy: 0.5582 - val_loss: 1.5031 - val_accuracy: 0.5113	20

(img_size, img_size, 3))
# vgg16_model.summary() 	
 	
vgg16_model_touse = tf.keras.models.Model(inputs = vgg16_model.input, outputs= vgg16_model.get_layer('block3_pool').output)
# vgg16_model_touse.summary() 	
 	
model = tf.keras.Sequential()
model.add(vgg16_model_touse)
model.add(tf.keras.layers.Conv2D(128, (3, 3), padding = 'same', activation = 'relu'))
model.add(tf.keras.layers.MaxPool2D(pool_size = (2, 2)))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPool2D((2, 2)))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(dropout))
model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))		

vgg16_model = tf.keras.applications.vgg16.VGG16(include_top = False, weights = 'imagenet', pooling = 'maxpooling', input_shape = (img_size, img_size, 3))
 	
vgg16_model_touse = tf.keras.models.Model(inputs = vgg16_model.input, outputs= vgg16_model.get_layer('block3_pool').output)
	loss: 1.2612 - accuracy: 0.5559 - val_loss: 1.4089 - val_accuracy: 0.5243	15

```
model = Sequential()
model.add(vgg16_model_touse)

model.add(Conv2D(filters=32,
kernel_size=(3,3),
padding='same',
            activation='relu',
input_shape=(img_size, img_size,
3)))

model.add(Conv2D(32, (3,3),
activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3,3),
padding='same', activation='relu'))
model.add(Conv2D(64, (3,3),
activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(128, (3,3),
padding='same', activation='relu'))
model.add(Conv2D(128, (3,3),
activation='relu'))
model.add(Activation('relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.2))


model.add(Flatten())
model.add(Dense(1024,
activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(11,
activation='softmax'))

model.summary()
model.layers[0].trainable = False
)
```

vgg16_model = tf.keras.applications.vgg16.VGG16(include_top = False, weights = 'imagenet', pooling = 'maxpooling', input_shape = (img_size, img_size, 3))
vgg16_model_touse = tf.keras.models.Model(inputs = vgg16_model.input, outputs= vgg16_model.get_layer('block3_pool').output)
 	
model = Sequential()
model.add(vgg16_model_touse)
model.add(Conv2D(filters=128, kernel_size=(3,3), padding='same',
activation='relu', input_shape=(img_size, img_size, 3)))
 	
model.add(Conv2D(256, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
#model.add(Dropout(0.2)) # Dropout
 	
model.add(Conv2D(512, (3,3), padding='same', activation='relu'))
model.add(Conv2D(512, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
 	
model.add(Conv2D(512, (3,3), padding='same', activation='relu'))
model.add(Conv2D(512, (3,3), activation='relu'))
model.add(Activation('relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.2))
	loss: 0.7833 - accuracy: 0.7309 - val_loss: 1.7495 - val_accuracy: 0.5353	20

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(11, activation='softmax'))		

vgg16_model = tf.keras.applications.vgg16.VGG16(include_top = False, weights = 'imagenet', pooling = 'maxpooling', input_shape = (img_size, img_size, 3))
 	
vgg16_model_touse = tf.keras.models.Model(inputs = vgg16_model.input, outputs= vgg16_model.get_layer('block3_pool').output)
# vgg16_model_touse.summary() 	
 	
model = tf.keras.Sequential()
model.add(vgg16_model_touse)
model.add(tf.keras.layers.Conv2D(128, (3, 3), padding = 'same', activation = 'relu'))
model.add(tf.keras.layers.MaxPool2D(pool_size = (2, 2)))
model.add(tf.keras.layers.GlobalMaxPool2D())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
 	
model.summary()
model.layers[0].trainable = False
	loss: 1.1574 - accuracy: 0.6114 - val_loss: 2.0511 - val_accuracy: 0.4264	20

num_classes = 11
model = Sequential()
resnet_model =
	loss: 0.0124 - accuracy: 1.0000 - val_loss: 0.8515 - val_accuracy: 0.7304	30

tf.keras.applications.resnet50.ResNet50(input_shape = (img_size,img_size,3), weights="imagenet", include_top = False, pooling = 'avg')
model.add(resnet_model)
model.add(Dense(num_classes, activation = 'softmax'))		