In [1]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)

#   train_datagen = ImageDataGenerator(
#           rescale=1./255,
#           shear_range=0.2,
#           zoom_range=0.2,
#           horizontal_flip=True)

#   datagen = ImageDataGenerator(
#           rotation_range=40,
#           width_shift_range=0.2,
#           height_shift_range=0.2,
#           rescale=1./255,
#           shear_range=0.2,
#           zoom_range=0.2,
#           horizontal_flip=True,
#           fill_mode='nearest')

# Flow training images in batches of 128 using train_datagen generator
train_generator = train_datagen.flow_from_directory(  # train_datagen
        'NEU_DET/Train_IMAGES',  # This is the source directory for training images
        target_size=(200,200),  # All images will be resized to 200 x 200
        color_mode='grayscale',
        batch_size = 64,
        # Specify the classes explicitly
        classes = ['crazing','inclusion','patches','pitted_surface','rolled_in_scale','scratches'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')


# All images will be rescaled by 1./255
test_datagen = ImageDataGenerator(rescale=1./255)

# Flow training images in batches of 128 using train_datagen generator
test_generator = test_datagen.flow_from_directory(
        'NEU_DET/Test_IMAGES',  # This is the source directory for training images
        target_size=(200,200),  # All images will be resized to 200 x 200
        color_mode='grayscale', # grayscale, rgb
        batch_size = 64,  # 128, 64, 32, 24
        # Specify the classes explicitly
        classes = ['crazing','inclusion','patches','pitted_surface','rolled_in_scale','scratches'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')
```

```
Found 1464 images belonging to 6 classes.
Found 336 images belonging to 6 classes.
```

In [2]:

```python
import tensorflow as tf

model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 1)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a dense layer
    tf.keras.layers.Flatten(),
    # 128 neuron in the fully-connected layer
    tf.keras.layers.Dense(1024, activation='relu'),  # 1024, 128, 64, 32
    # 6 output neurons for 6 classes with the softmax activation
    tf.keras.layers.Dense(6, activation='sigmoid')  # softmax
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 198, 198, 16)      160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 99, 99, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 97, 97, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 64)        36928
```

```
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              1049600
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 6150
=================================================================
Total params: 1,152,902
Trainable params: 1,152,902
Non-trainable params: 0
_________________________________________________________________
```

In [3]:

```
#compile model using accuracy to measure model performance
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [4]:

```
history = model.fit(
        train_generator,
        validation_data = test_generator,
        epochs = 30)
```

```
Epoch 1/30
23/23 [==============================] - 72s 2s/step - loss: 1.7744 - accuracy: 0.2018 - val_loss: 1.5662 - val_accuracy: 0.3512
Epoch 2/30
23/23 [==============================] - 38s 2s/step - loss: 1.2816 - accuracy: 0.4827 - val_loss: 0.9700 - val_accuracy: 0.6577
Epoch 3/30
23/23 [==============================] - 41s 2s/step - loss: 0.6345 - accuracy: 0.7676 - val_loss: 0.4703 - val_accuracy: 0.8036
Epoch 4/30
23/23 [==============================] - 32s 1s/step - loss: 0.3978 - accuracy: 0.8611 - val_loss: 0.3980 - val_accuracy: 0.8452
Epoch 5/30
23/23 [==============================] - 31s 1s/step - loss: 0.3343 - accuracy: 0.8791 - val_loss: 0.3727 - val_accuracy: 0.8780
Epoch 6/30
23/23 [==============================] - 31s 1s/step - loss: 0.3609 - accuracy: 0.8703 - val_loss: 0.4429 - val_accuracy: 0.8512
Epoch 7/30
23/23 [==============================] - 33s 1s/step - loss: 0.2678 - accuracy: 0.9028 - val_loss: 0.1820 - val_accuracy: 0.9375
Epoch 8/30
23/23 [==============================] - 36s 2s/step - loss: 0.2038 - accuracy: 0.9156 - val_loss: 0.2683 - val_accuracy: 0.9435
Epoch 9/30
23/23 [==============================] - 40s 2s/step - loss: 0.2176 - accuracy: 0.9144 - val_loss: 0.1830 - val_accuracy: 0.9405
Epoch 10/30
23/23 [==============================] - 39s 2s/step - loss: 0.1697 - accuracy: 0.9385 - val_loss: 0.2528 - val_accuracy: 0.8929
Epoch 11/30
```

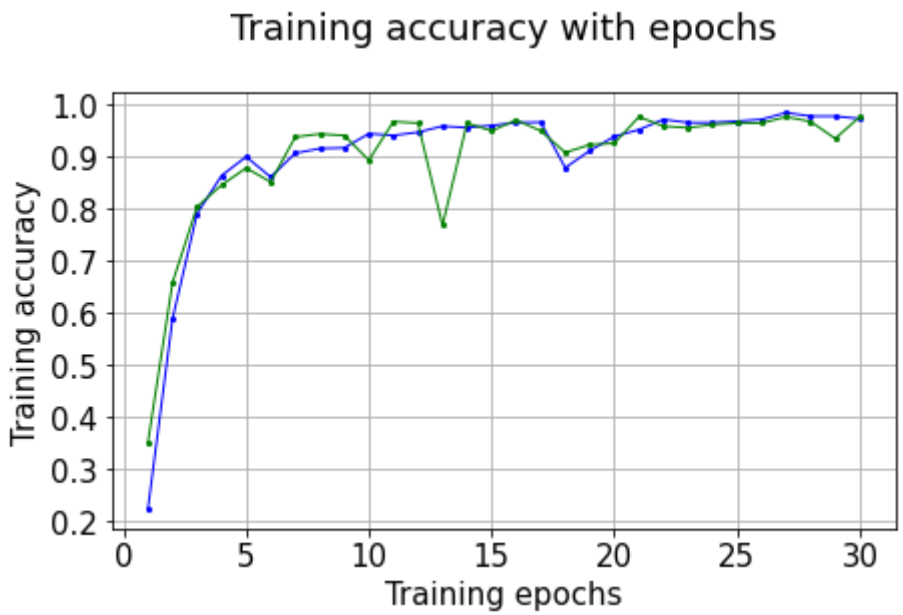
```
23/23 [==============================] - 37s 2s/step - loss: 0.1454 - accuracy: 0.9407 - val_loss: 0.1917 - val_accuracy: 0.9673
Epoch 12/30
23/23 [==============================] - 37s 2s/step - loss: 0.1251 - accuracy: 0.9485 - val_loss: 0.1403 - val_accuracy: 0.9643
Epoch 13/30
23/23 [==============================] - 37s 2s/step - loss: 0.0961 - accuracy: 0.9642 - val_loss: 0.5133 - val_accuracy: 0.7679
Epoch 14/30
23/23 [==============================] - 37s 2s/step - loss: 0.1376 - accuracy: 0.9467 - val_loss: 0.1058 - val_accuracy: 0.9643
Epoch 15/30
23/23 [==============================] - 40s 2s/step - loss: 0.0957 - accuracy: 0.9600 - val_loss: 0.1739 - val_accuracy: 0.9494
Epoch 16/30
23/23 [==============================] - 36s 2s/step - loss: 0.0849 - accuracy: 0.9682 - val_loss: 0.1341 - val_accuracy: 0.9702
Epoch 17/30
23/23 [==============================] - 38s 2s/step - loss: 0.0784 - accuracy: 0.9672 - val_loss: 0.1338 - val_accuracy: 0.9494
Epoch 18/30
23/23 [==============================] - 36s 2s/step - loss: 0.2974 - accuracy: 0.9071 - val_loss: 0.3220 - val_accuracy: 0.9077
Epoch 19/30
23/23 [==============================] - 46s 2s/step - loss: 0.2598 - accuracy: 0.9137 - val_loss: 0.2519 - val_accuracy: 0.9226
Epoch 20/30
23/23 [==============================] - 46s 2s/step - loss: 0.1514 - accuracy: 0.9405 - val_loss: 0.2321 - val_accuracy: 0.9256
Epoch 21/30
23/23 [==============================] - 42s 2s/step - loss: 0.1333 - accuracy: 0.9426 - val_loss: 0.0937 - val_accuracy: 0.9762
Epoch 22/30
23/23 [==============================] - 46s 2s/step - loss: 0.0791 - accuracy: 0.9742 - val_loss: 0.1364 - val_accuracy: 0.9583
Epoch 23/30
23/23 [==============================] - 33s 1s/step - loss: 0.0706 - accuracy: 0.9792 - val_loss: 0.1425 - val_accuracy: 0.9554
Epoch 24/30
23/23 [==============================] - 32s 1s/step - loss: 0.0933 - accuracy: 0.9561 - val_loss: 0.1266 - val_accuracy: 0.9613
Epoch 25/30
23/23 [==============================] - 31s 1s/step - loss: 0.0870 - accuracy: 0.9654 - val_loss: 0.1240 - val_accuracy: 0.9643
Epoch 26/30
23/23 [==============================] - 32s 1s/step - loss: 0.0649 - accuracy: 0.9723 - val_loss: 0.1049 - val_accuracy: 0.9643
Epoch 27/30
23/23 [==============================] - 31s 1s/step - loss: 0.0600 - accuracy: 0.9830 - val_loss: 0.0898 - val_accuracy: 0.9762
Epoch 28/30
23/23 [==============================] - 32s 1s/step - loss: 0.0564 - accuracy: 0.9782 - val_loss: 0.1208 - val_accuracy: 0.9673
Epoch 29/30
23/23 [==============================] - 32s 1s/step - loss: 0.0611 - accuracy: 0.9725 - val_loss: 0.1757 - val_accuracy: 0.9345
Epoch 30/30
23/23 [==============================] - 31s 1s/step - loss: 0.0558 - accuracy: 0.9833 - val_loss: 0.1022 - val_accuracy: 0.9762
```

In [5]:

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(7,4))
plt.plot([i+1 for i in range(30)],history.history['accuracy'],'-o',c='b',lw=1,markersize=2)
plt.plot([i+1 for i in range(30)],history.history['val_accuracy'],'-o',c='g',lw=1,markersize=2)
plt.grid(True)
plt.title("Training accuracy with epochs\n",fontsize=18)
plt.xlabel("Training epochs",fontsize=15)
plt.ylabel("Training accuracy",fontsize=15)
plt.xticks(fontsize=15)
```

```
plt.yticks(fontsize=15)
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: