In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
import os

from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,Dense,MaxPooling2D,Activation,Dropout,BatchNorma
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
```

In [4]:

```
train = ImageDataGenerator(rescale=1/255)
test = ImageDataGenerator(rescale=1/255)
train_dataset = train.flow_from_directory(directory='C:/Users/HP/OneDrive/Desktop/seg_train

test_dataset = test.flow_from_directory(directory='C:/Users/HP/OneDrive/Desktop/seg_test/se
```

```
Found 14034 images belonging to 6 classes.
Found 3000 images belonging to 6 classes.
```

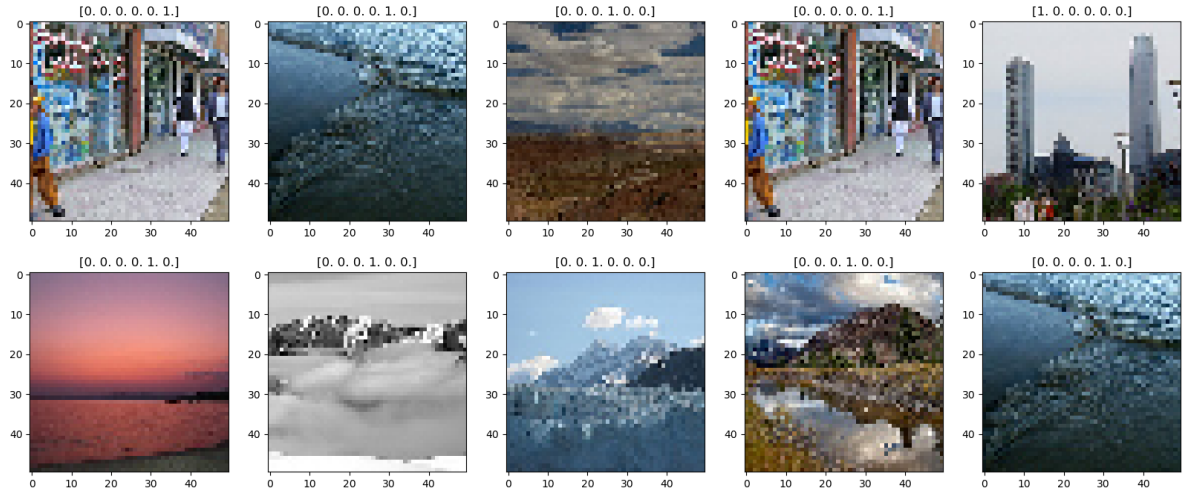
In [5]:

```
indices = [np.random.randint(32) for i in range(10)]
print(indices)

plt.figure(figsize=(20,8))
for i in enumerate(indices):
    plt.subplot(2,5,i[0]+1)
    plt.imshow(train_dataset[0][0][i[1]])
    plt.title(train_dataset[0][1][i[1]])
plt.show()
```

```
[1, 19, 18, 1, 27, 30, 10, 4, 17, 19]

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\text.py:1223: FutureWa
rning: elementwise comparison failed; returning scalar instead, but in the f
uture will perform elementwise comparison
  if s != self._text:
```



In [6]:

```
values = list(train_dataset.class_indices.values())
keys = list(train_dataset.class_indices.keys())


dics = list(map(lambda x,y:{x:y},values,keys))

from functools import reduce
mappings = reduce(lambda x,y:{**x,**y},dics)
mappings
#print(values)
#print(keys)
#print(dics)
```

Out[6]:

```
{0: 'buildings',
 1: 'forest',
 2: 'glacier',
 3: 'mountain',
 4: 'sea',
 5: 'street'}
```

In [7]:

```
model = Sequential()
model.add(Conv2D(filters=32,kernel_size=(3,3),padding='same',input_shape=(50,50,3)))
model.add(Activation('relu'))
model.add(Conv2D(filters=32,kernel_size=(3,3)))
model.add(Dropout(0.25))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(filters=50,kernel_size=(3,3),padding='same',input_shape=(50,50,3)))
model.add(Activation('relu'))
model.add(Conv2D(filters=50,kernel_size=(3,3)))
model.add(Dropout(0.25))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(filters=75,kernel_size=(3,3),padding='same',input_shape=(50,50,3)))
model.add(Activation('relu'))
model.add(Conv2D(filters=75,kernel_size=(3,3)))
model.add(Dropout(0.25))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
kernel_regularizer = keras.regularizers.l1_l2(l1=1e-5,l2=1e-4)
model.add(Dense(units=50,activation='relu',kernel_regularizer=kernel_regularizer))
model.add(Dense(50,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(6,activation='softmax'))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 50, 50, 32)        896

 activation (Activation)     (None, 50, 50, 32)        0

 conv2d_1 (Conv2D)           (None, 48, 48, 32)        9248

 dropout (Dropout)           (None, 48, 48, 32)        0

 activation_1 (Activation)   (None, 48, 48, 32)        0

 max_pooling2d (MaxPooling2D  (None, 24, 24, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 24, 24, 50)        14450

 activation_2 (Activation)   (None, 24, 24, 50)        0

 conv2d_3 (Conv2D)           (None, 22, 22, 50)        22550

 dropout_1 (Dropout)         (None, 22, 22, 50)        0
```

```
 activation_3 (Activation)   (None, 22, 22, 50)        0

 max_pooling2d_1 (MaxPooling  (None, 11, 11, 50)       0
 2D)

 conv2d_4 (Conv2D)           (None, 11, 11, 75)        33825

 activation_4 (Activation)   (None, 11, 11, 75)        0

 conv2d_5 (Conv2D)           (None, 9, 9, 75)          50700

 dropout_2 (Dropout)         (None, 9, 9, 75)          0

 activation_5 (Activation)   (None, 9, 9, 75)          0

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 75)         0
 2D)

 flatten (Flatten)           (None, 1200)              0

 dense (Dense)               (None, 50)                60050

 dense_1 (Dense)             (None, 50)                2550

 dropout_3 (Dropout)         (None, 50)                0

 dense_2 (Dense)             (None, 6)                 306

=================================================================
Total params: 194,575
Trainable params: 194,575
Non-trainable params: 0
_________________________________________________________________
```

In [8]:

```
model.compile(loss='CategoricalCrossentropy',optimizer = 'adam',metrics='accuracy')
history = model.fit(train_dataset,batch_size=80,epochs=2,validation_data=test_dataset)
```

```
Epoch 1/2
439/439 [==============================] - 63s 141ms/step - loss: 1.1648 - a
ccuracy: 0.5379 - val_loss: 1.0766 - val_accuracy: 0.6370
Epoch 2/2
439/439 [==============================] - 70s 159ms/step - loss: 0.9480 - a
ccuracy: 0.6298 - val_loss: 1.0103 - val_accuracy: 0.6287
```

In [9]:

```
plt.plot(history.history['accuracy'],label='accuracy')
plt.plot(history.history['val_accuracy'],label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
```

Out[9]:

```
<matplotlib.legend.Legend at 0x139a15a3ac0>
```