

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

```
# %reset
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

Imports

```
!pip install ipython-autotime
```

Collecting ipython-autotime

Downloading <https://files.pythonhosted.org/packages/d6/c5/013f5aa3b56c6d2c58634bc97977/>
 Requirement already satisfied: ipython in /usr/local/lib/python3.6/dist-packages (from ipython-autotime==0.3.0)
 Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: simplegeneric>=0.8 in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-packages (from ipython==5.1.0)

 Installing collected packages: ipython-autotime

 Successfully installed ipython-autotime-0.3.0

```
# necessary imports
import os
import cv2
import numpy as np
from imutils import paths
from sklearn.preprocessing import LabelBinarizer
from tqdm import tqdm

import matplotlib.pyplot as plt
%matplotlib inline

from google.colab.patches import cv2_imshow
```

```
%load_ext autotime
```

```
time: 122 µs (started: 2021-01-08 07:15:14 +00:00)
```

▼ Initializing

```
img_width = 50
img_height = 50
nb_epochs = 25
batch_siz = 32
```

```
time: 1.04 ms (started: 2021-01-08 07:21:31 +00:00)
```

▼ VGG Model

```
from keras.applications.vgg16 import VGG16
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten
```

```
time: 1.38 s (started: 2021-01-08 01:35:33 +00:00)
```

```
# load VGG16 model without classification layers
model = VGG16(include_top=False, input_shape=(img_width, img_height, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/58892288/58889256 [=====] - 1s 0us/step
time: 6.79 s (started: 2021-01-08 01:35:34 +00:00)
```



```
# add new classification layers
flat1 = Flatten()(model.layers[-1].output) # flatten last layer
class1 = Dense(1024, activation='relu')(flat1) # add FC layer on previous layer
output = Dense(6, activation='softmax')(class1) # add softmax layer
```

```
time: 20.8 ms (started: 2021-01-08 01:35:41 +00:00)
```

```
# define the new model
model = Model(inputs=model.inputs, outputs=output)
model.summary()
```

```
Model: "model"
```

| Layer (type) | Output Shape | Param # |
|--------------|--------------|---------|
| ===== | | |

| | | |
|----------------------------|---------------------|---------|
| input_1 (InputLayer) | [(None, 50, 50, 3)] | 0 |
| block1_conv1 (Conv2D) | (None, 50, 50, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, 50, 50, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, 25, 25, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 25, 25, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, 25, 25, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, 12, 12, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 12, 12, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, 12, 12, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, 12, 12, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, 6, 6, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 6, 6, 512) | 1180160 |
| block4_conv2 (Conv2D) | (None, 6, 6, 512) | 2359808 |
| block4_conv3 (Conv2D) | (None, 6, 6, 512) | 2359808 |
| block4_pool (MaxPooling2D) | (None, 3, 3, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 3, 3, 512) | 2359808 |
| block5_conv2 (Conv2D) | (None, 3, 3, 512) | 2359808 |
| block5_conv3 (Conv2D) | (None, 3, 3, 512) | 2359808 |
| block5_pool (MaxPooling2D) | (None, 1, 1, 512) | 0 |
| flatten (Flatten) | (None, 512) | 0 |
| dense (Dense) | (None, 1024) | 525312 |
| dense_1 (Dense) | (None, 6) | 6150 |

=====

Total params: 15,246,150
Trainable params: 15,246,150
Non-trainable params: 0

time: 14.6 ms (started: 2021-01-08 01:35:41 +00:00)

Compile the model

```
from keras.optimizers import SGD
sgd = SGD(lr=0.001, decay=1e-7, momentum=.9)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
```

time: 20.1 ms (started: 2021-01-08 01:35:41 +00:00)

▼ Loading Data

```
# Cutout Function
def apply_mask(image, size=12, n_squares=1):
    h, w, channels = image.shape
    new_image = image
    y = np.random.randint(h)
    x = np.random.randint(w)
    y1 = np.clip(y - size // 2, 0, h)
    y2 = np.clip(y + size // 2, 0, h)
    x1 = np.clip(x - size // 2, 0, w)
    x2 = np.clip(x + size // 2, 0, w)
    new_image[y1:y2,x1:x2,:] = 0
    return new_image
```

time: 4.77 ms (started: 2021-01-08 01:35:41 +00:00)

```
# A function to load data from a given directory
def load_cutout_data(data_dir):
    data = []
    labels = []
    class_dirs = os.listdir(data_dir)

    for direc in class_dirs:
        class_dir = os.path.join(data_dir, direc)
        for imagepath in tqdm(list(paths.list_images(class_dir))):
            image = cv2.imread(imagepath)
            image = cv2.resize(image, (img_width, img_height)) # incase images not of same size
            image = apply_mask(image)
            data.append(image)
            labels.append(direc)
    # normalizing and converting to numpy array format
    data = np.array(data, dtype='float')/255.0
    labels = np.array(labels)
    return data, labels
```

time: 10.3 ms (started: 2021-01-08 01:35:41 +00:00)

```
# A function to load data from a given directory
def load_data(data_dir):
    data = []
    labels = []
    class_dirs = os.listdir(data_dir)

    for direc in class_dirs:
```

```

class_dir = os.path.join(data_dir, direc)
for imagepath in tqdm(list(paths.list_images(class_dir))):
    image = cv2.imread(imagepath)
    image = cv2.resize(image, (img_width, img_height)) # incase images not of same size
    data.append(image)
    labels.append(direc)
# normalizing and converting to numpy array format
data = np.array(data, dtype='float')/255.0
labels = np.array(labels)
return data, labels

```

time: 15.6 ms (started: 2021-01-08 07:20:15 +00:00)

```

train_dir = "/content/drive/MyDrive/CV/Assignment 3/seg_train/seg_train/"
test_dir = "/content/drive/MyDrive/CV/Assignment 3/seg_test/seg_test/"
pred_dir = "/content/drive/MyDrive/CV/Assignment 3/pred/seg_pred/seg_pred/"

```

time: 1.02 ms (started: 2021-01-08 07:15:54 +00:00)

```

print('loading train images')
X_train, y_train = load_cutout_data(train_dir)

```

```

loading train images
100%|██████████| 2190/2190 [06:58<00:00, 5.23it/s]
100%|██████████| 2271/2271 [06:33<00:00, 5.77it/s]
100%|██████████| 2404/2404 [07:01<00:00, 5.71it/s]
100%|██████████| 2512/2512 [07:24<00:00, 5.65it/s]
100%|██████████| 2274/2274 [06:50<00:00, 5.54it/s]
100%|██████████| 2382/2382 [07:27<00:00, 5.32it/s]
time: 42min 28s (started: 2021-01-08 01:35:41 +00:00)

```

```

X_valid, y_valid = load_data(test_dir)

```

```

100%|██████████| 437/437 [01:31<00:00, 4.76it/s]
100%|██████████| 474/474 [01:28<00:00, 5.37it/s]
100%|██████████| 553/553 [01:52<00:00, 4.92it/s]
100%|██████████| 525/525 [01:33<00:00, 5.63it/s]
100%|██████████| 510/510 [01:27<00:00, 5.82it/s]
100%|██████████| 501/501 [01:29<00:00, 5.60it/s]
time: 9min 25s (started: 2021-01-08 02:18:10 +00:00)

```

```

X_train = np.append(X_train, X_valid, axis=0)
y_train = np.append(y_train, y_valid, axis=0)

```

time: 412 ms (started: 2021-01-08 02:27:35 +00:00)

```

lb = LabelBinarizer()

```

time: 1.02 ms (started: 2021-01-08 08:17:45 +00:00)

```
y_train = lb.fit_transform(y_train)
```

```
time: 22.6 ms (started: 2021-01-08 02:27:36 +00:00)
```

```
from sklearn.model_selection import train_test_split
(X_train, X_valid, y_train, y_valid) = train_test_split(X_train, y_train, test_size=0.2, rand
```

```
time: 291 ms (started: 2021-01-08 02:27:36 +00:00)
```

Train the model

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
```

```
time: 968 µs (started: 2021-01-08 02:27:36 +00:00)
```

```
# patient early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=2)
mc = ModelCheckpoint('best_model.h5', monitor='val_accuracy', mode='max', verbose=1, save_bes
```

```
time: 1.84 ms (started: 2021-01-08 02:27:36 +00:00)
```

```
H = model.fit(X_train, y_train, batch_size=batch_siz,
              epochs=nb_epochs,
              validation_data=(X_valid, y_valid),
              verbose=0, callbacks=[es, mc])
```

```
Epoch 00001: val_accuracy improved from -inf to 0.84444, saving model to best_model.h5
```

```
Epoch 00002: val_accuracy improved from 0.84444 to 0.85060, saving model to best_model.h5
```

```
Epoch 00003: val_accuracy improved from 0.85060 to 0.86264, saving model to best_model.h5
```

```
Epoch 00004: val_accuracy improved from 0.86264 to 0.88113, saving model to best_model.h5
```

```
Epoch 00005: val_accuracy did not improve from 0.88113
```

```
Epoch 00006: val_accuracy improved from 0.88113 to 0.88230, saving model to best_model.h5
```

```
Epoch 00006: early stopping
```

```
time: 2min (started: 2021-01-08 02:27:36 +00:00)
```



```
save_path = '/content/drive/MyDrive/CV/Assignment 3/vgg16_sgd_cutout'
```

```
time: 981 µs (started: 2021-01-08 02:29:37 +00:00)
```

```
# save the model's trained weights
model.save_weights(save_path+"transfer_trained_wts.h5")
```

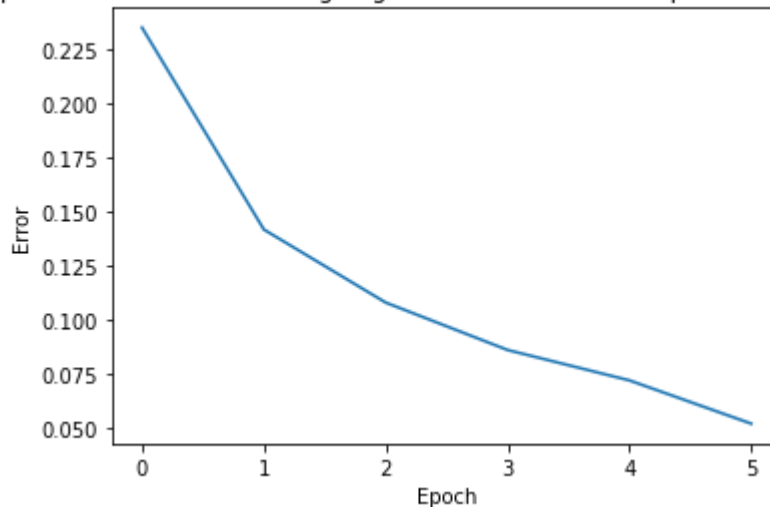
time: 228 ms (started: 2021-01-08 02:29:37 +00:00)

```
# model.load_weights('/content/drive/MyDrive/CV/Assignment 3/vgg_aug_transfer_trained_wts.h5')
```

time: 561 μ s (started: 2021-01-08 02:29:37 +00:00)

```
simple_acc = H.history['accuracy']
plt.plot([1 - acc for acc in simple_acc])
plt.title('Error for a InceptionResNetV2 model using augmentation with Adam optimizer & adaptive learning rate')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.savefig(save_path+'/simple_acc_error.png')
plt.show()
```

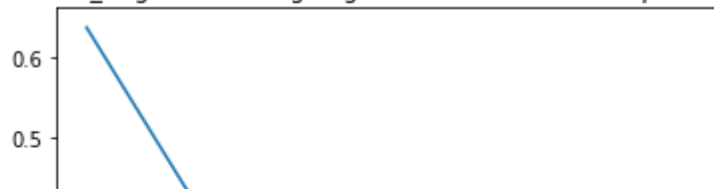
Error for a InceptionResNetV2 model using augmentation with Adam optimizer & adaptive learning rate



time: 236 ms (started: 2021-01-08 02:29:37 +00:00)

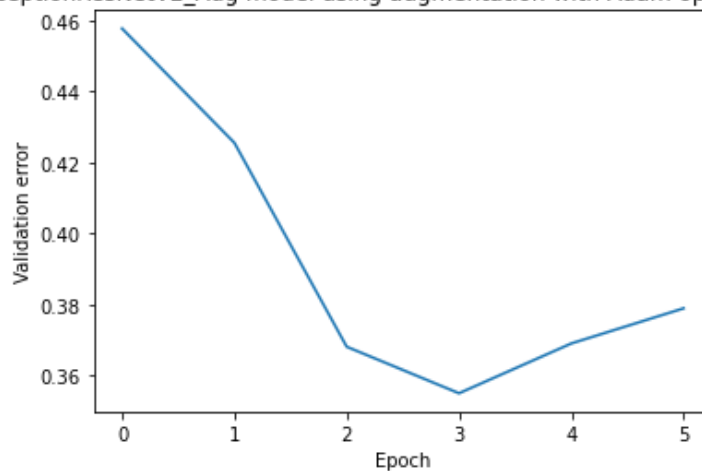
```
simple_loss = H.history['loss']
plt.plot([los for los in simple_loss])
plt.title('Loss for a InceptionResNetV2_Aug model using augmentation with Adam optimizer & adaptive learning rate')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.savefig(save_path+'/simple_loss_error.png')
plt.show()
```

Loss for a InceptionResNetV2_Aug model using augmentation with Adam optimizer & adaptive learning rate



```
simple_val_loss = H.history['val_loss']
plt.plot([los for los in simple_val_loss])
plt.title('Validation Loss for a InceptionResNetV2_Aug model using augmentation with Adam opt
plt.ylabel('Validation error')
plt.xlabel('Epoch')
plt.savefig(save_path+'/simple_Validation_loss_error.png')
plt.show()
```

Validation Loss for a InceptionResNetV2_Aug model using augmentation with Adam optimizer & adaptive learning rate



time: 213 ms (started: 2021-01-08 02:29:38 +00:00)

```
simple_val_acc = H.history['val_accuracy']
plt.plot([1 - acc for acc in simple_val_acc])
plt.title('Validation error for a InceptionResNetV2_Aug model using augmentation with Adam op
plt.ylabel('Validation error')
plt.xlabel('Epoch')
plt.savefig(save_path+'/simple_Validation_error.png')
plt.show()
```


Validation error for a InceptionResNetV2_Aug model using augmentation with Adam optimizer & adaptive learning rate



```
print('loading test images')
X_test, y_test = load_data(pred_dir)
```

```
0%|          | 0/1330 [00:00<?, ?it/s]loading test images
100%|██████████| 1330/1330 [06:24<00:00, 3.46it/s]
100%|██████████| 1297/1297 [06:16<00:00, 3.44it/s]
100%|██████████| 1128/1128 [05:22<00:00, 3.50it/s]
100%|██████████| 1166/1166 [05:45<00:00, 3.37it/s]
100%|██████████| 1236/1236 [05:56<00:00, 3.47it/s]
100%|██████████| 1144/1144 [05:31<00:00, 3.45it/s]
```

NameError Traceback (most recent call last)

<ipython-input-11-85728dc5b043> in <module>()

```
1 print('loading test images')
2 X_test, y_test = load_data(pred_dir)
----> 3 y_test = lb.fit_transform(y_test)
```

NameError: name 'lb' is not defined

SEARCH STACK OVERFLOW

time: 35min 17s (started: 2021-01-08 07:21:40 +00:00)

```
y_test = lb.fit_transform(y_test)
```

time: 13.7 ms (started: 2021-01-08 08:18:05 +00:00)

```
score = model.evaluate(X_test, y_test, batch_size=batch_size)
print('Test Loss = ', score[0])
print('Test Accuracy = ', score[1])
```

```
'''CONFUSION MATRIX'''
# Making prediction
y_pred = model.predict(X_test)
y_true = np.argmax(y_test, axis=-1)

# Plotting the confusion matrix
from sklearn.metrics import confusion_matrix
confusion_mtx = confusion_matrix(y_true, np.argmax(y_pred, axis=1))
```

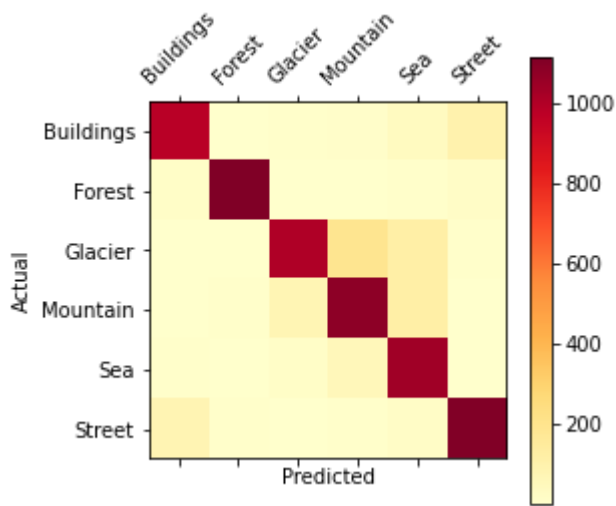
time: 2.72 s (started: 2021-01-08 02:59:42 +00:00)

```
confusion_mtx
```

```
array([[ 983,    3,    9,   12,   38,   99],
```

```
[ 16, 1117, 3, 2, 6, 22],
[ 1, 5, 1005, 194, 119, 6],
[ 5, 6, 76, 1086, 120, 4],
[ 7, 2, 15, 58, 1041, 5],
[ 82, 8, 3, 9, 22, 1112]]time: 3.87 ms (started: 2021-01-08 02:59:45)
```

```
def plot_confusion_matrix(df_confusion, title='Confusion matrix', cmap=plt.cm.YlOrRd):
    plt.matshow(df_confusion, cmap=cmap) # imshow
    plt.colorbar()
    tick_marks = np.arange(6)
    names = ["Buildings", "Forest", "Glacier", "Mountain", "Sea", "Street"]
    plt.xticks(tick_marks, names, rotation=45)
    plt.yticks(tick_marks, names)
    plt.ylabel("Actual")
    plt.xlabel("Predicted")
#call function
plot_confusion_matrix(confusion_mtx)
```



time: 167 ms (started: 2021-01-08 02:59:45 +00:00)

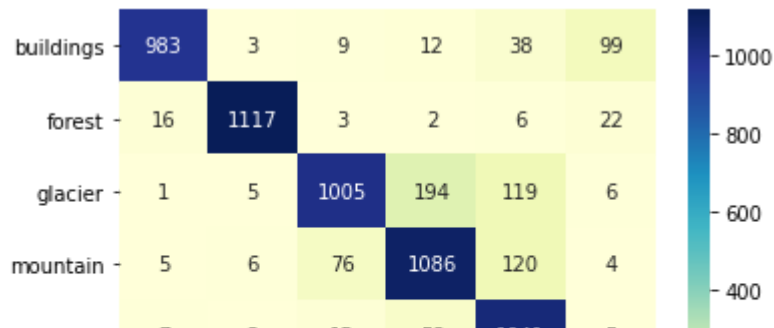
```
import seaborn as sns

class_names = ['buildings','street','forest','glacier','mountain','sea']

class_names = sorted(class_names)

sns.heatmap(confusion_mtx, xticklabels=class_names, yticklabels=class_names,
            annot=True, fmt='d', cmap="YlGnBu")
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f69ba191128>



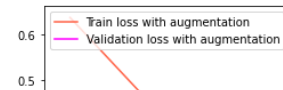
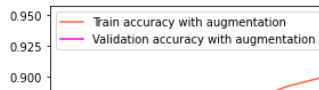
```
fig, axis = plt.subplots(1, 2, figsize=(20, 4))
```

```
axis[0].plot(H.history['accuracy'],
             label='Train accuracy with augmentation',
             c='tomato', ls='-')
axis[0].plot(H.history['val_accuracy'],
             label='Validation accuracy with augmentation',
             c='magenta', ls='-')
```

```
axis[0].set_xlabel('Epoch')
axis[0].set_ylabel('Accuracy')
axis[0].legend(loc='upper left')
```

```
axis[1].plot(H.history['loss'],
             label='Train loss with augmentation',
             c='tomato', ls='-')
axis[1].plot(H.history['val_loss'],
             label='Validation loss with augmentation',
             c='magenta', ls='-')
```

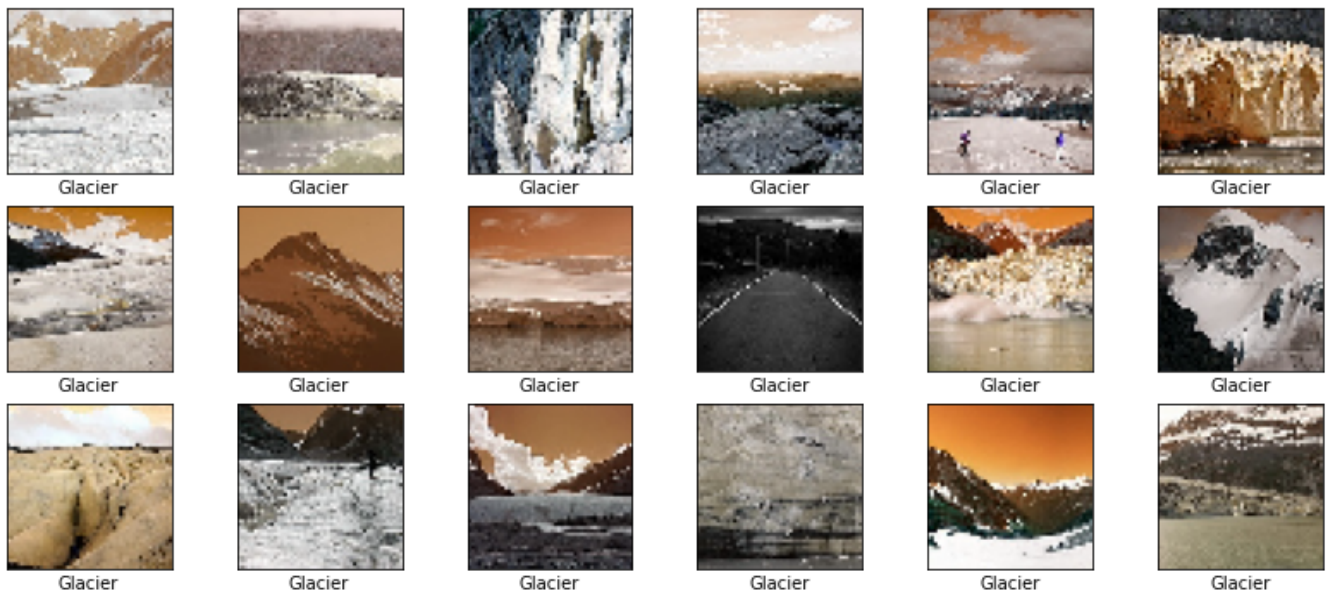
```
axis[1].set_xlabel('Epoch')
axis[1].set_ylabel('loss')
axis[1].legend(loc='upper left')
plt.savefig(save_path+'simple_Validation_error&loss.png')
plt.show()
```



```
def visualize_data(images, categories, class_names):
    fig = plt.figure(figsize=(14, 6))
    fig.patch.set_facecolor('white')
    already = []
    for i in range(3 * 6):
        if
            plt.subplot(3, 6, i+1)
            plt.xticks([])
            plt.yticks([])
            plt.imshow(images[i])
            already.append(images[i])
            class_index = categories[i].argmax()
            plt.xlabel(class_names[class_index])
    plt.show()
```

time: 6.46 ms (started: 2021-01-08 08:19:10 +00:00)

```
names = ["Buildings", "Forest", "Glacier", "Mountain", "Sea", "Street"]
visualize_data(X_test, y_test, names)
```



time: 620 ms (started: 2021-01-08 08:19:13 +00:00)

