AITEX FABRIC IMAGE DATABASE

- https://www.aitex.es/afid/ (https://www.aitex.es/afid/)

Data

- The textile fabric database consists of 245 images of 7 different fabrics
- Images have a size of 4096×256 pixels
- There are 140 defect-free images, 20 for each type of fabric
- With different types of defects, there are 105 images

Environment

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

In [ ]:
```
% cd ./drive/MyDrive/colab_notebook/image/
```

```
/content/drive/MyDrive/colab_notebook/image
```

Data set

In [ ]:
```
! pip install opencv-python
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.
7/dist-packages (4.1.2.30)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.
7/dist-packages (from opencv-python) (1.19.5)
```

In [ ]:
```
import cv2
import os
import glob
import shutil
import random
import string
import numpy as np
```

In [ ]:
```
PATH_DEFECT = 'dataset/Defect_images/'
PATH_MASK = 'dataset/Mask_images/'
PATH_NODEFECT = 'dataset/NODefect_images/'
```

In [ ]:
```
random.seed(0)

defect_list = glob.glob(PATH_DEFECT + '*.png')
mask_list = glob.glob(PATH_MASK + '*.png')
pass_list = glob.glob(PATH_NODEFECT + '**/*.png')

# Match defect-mask pairs
new_defect_list = list()
new_mask_list = list()
for defect in defect_list:
    num = defect.split('/')[-1].split('_')[0]
    for mask in mask_list:
        num_mask = mask.split('/')[-1].split('_')[0]
        if num == num_mask:
            new_defect_list.append(defect)
            new_mask_list.append(mask)
            break
defect_list = new_defect_list
mask_list = new_mask_list
```

In [ ]:
```
# train dataset
if os.path.exists('dataset/OK') is False:
    os.mkdir('dataset/OK')
if os.path.exists('dataset/FAIL') is False:
    os.mkdir('dataset/FAIL')
if os.path.exists('dataset/MASK') is False:
    os.mkdir('dataset/MASK')

idx = 0
for file_name in pass_list:
    img = cv2.imread(file_name)
    height, width, _ = img.shape
    step = height // 2

    for i in range(width // step):
        w = i * step
        if w < width - height and random.randint(0, 9) < 3:
            patch = img[:, w:w+height, :]
            cv2.imwrite('dataset/OK/%04d.png' % idx, patch)
            idx += 1

patch_pair_list = list()

for item in zip(defect_list, mask_list):
    defect, mask = item

    img_d = cv2.imread(defect)
    img_m = cv2.imread(mask)

    height, width, _ = img_d.shape
    step = height // 2
    for i in range(width // step):
        w = i * step
        if w < width - height:
            patch = img_d[:, w:w+height, :]
            patch_d = img_m[:, w:w+height, :]

            if patch_d.sum() > 0:
                patch_pair_list.append((patch, patch_d))

random.shuffle(patch_pair_list)
for idx, pair in enumerate(patch_pair_list):
    patch, patch_d = pair
    cv2.imwrite('dataset/FAIL/%04d.png' % idx, patch)
    cv2.imwrite('dataset/MASK/%04d.png' % idx, patch_d)
```

In [ ]:
```
# The test dataset
if os.path.exists('data/') is False:
    os.mkdir('data/')
if os.path.exists('tfrecords/') is False:
    os.mkdir('tfrecords/')
if os.path.exists('model/') is False:
    os.mkdir('model/')
if os.path.exists('data/input_data') is False:
    os.mkdir('data/input_data')
if os.path.exists('data/output_csv') is False:
    os.mkdir('data/output_csv')

idx = 0
for file_name in pass_list:
    img = cv2.imread(file_name)
    height, width, _ = img.shape
    step = height // 2

    for i in range(width // step):
        w = i * step
        if w < width - height and random.randint(0, 9) < 5:
            patch = img[:, w:w+height, :]
            cv2.imwrite('data/input_data/ok_%04d.png' % idx, patch)
            idx += 1

patch_pair_list = list()
for item in zip(defect_list, mask_list):
    defect, mask = item

    img_d = cv2.imread(defect)
    img_m = cv2.imread(mask)

    height, width, _ = img_d.shape
    step = height // 2
    for i in range(width // step):
        w = i * step
        if w < width - height:
            patch = img_d[:, w:w+height, :]
            patch_d = img_m[:, w:w+height, :]

            if patch_d.sum() > 0:
                patch_pair_list.append((patch, patch_d))

random.shuffle(patch_pair_list)
for idx, pair in enumerate(patch_pair_list):
    patch, patch_d = pair
    cv2.imwrite('data/input_data/fail_%04d.png' % idx, patch)
```

Data Preprocessing

- TFRecord Builder
    - Data Serialization to learn faster

In [ ]:
```
import glob
import os
import tensorflow as tf
import cv2
```

Paths and Hyperparameters

In [ ]:
```
DATASET_OK_PATTERN = 'dataset/OK/*.png'
DATASET_FAIL_PATTERN = 'dataset/FAIL/*.png'

# to serialize the data into binary
TFRECORD_PATH = 'tfrecords/'
IMAGE_PER_TFRECORD = 100
```

Import data

In [ ]:
```
ok_list = glob.glob(DATASET_OK_PATTERN)
fail_list = glob.glob(DATASET_FAIL_PATTERN)

num_ok = len(ok_list)
num_fail = len(fail_list)

# Oversampling
# to make the number of fail datas equal to number of ok datas
fail_list_new = list()
for _ in range(num_ok // num_fail):
    fail_list_new += fail_list
fail_list_new += fail_list[:num_ok % num_fail]
fail_list = fail_list_new

ok_label = [0] * len(ok_list)
fail_label = [1] * len(fail_list)

file_list = ok_list + fail_list
label_list = ok_label + fail_label
```

TFRecord functions

In [ ]:
```
def _bytes_feature(value):
    """Returns a bytes_list from a string / byte."""
    if isinstance(value, type(tf.constant(0))):
        value = value.numpy()
    return tf.train.Feature(bytes_list=tf.train.BytesList(value=[value
]))

def _int64_feature(value):
    """Returns an int64_list from a bool / enum / int / uint."""
    return tf.train.Feature(int64_list=tf.train.Int64List(value=[value
]))

def image_example(image_string, label):
    image_shape = tf.image.decode_image(image_string).shape

    feature = {
        'height': _int64_feature(image_shape[0]),
        'width': _int64_feature(image_shape[1]),
        'depth': _int64_feature(image_shape[2]),
        'label': _int64_feature(label),
        'image_raw': _bytes_feature(image_string),
    }

    return tf.train.Example(features=tf.train.Features(feature=feature))
```

Write TFRecords

In [ ]:
```
if os.path.exists(TFRECORD_PATH) is False:
    os.mkdir(TFRECORD_PATH)

num_tfrecords = len(file_list) // IMAGE_PER_TFRECORD
if len(file_list) % IMAGE_PER_TFRECORD != 0:
    num_tfrecords += 1

for idx in range(num_tfrecords):
    idx0 = idx * IMAGE_PER_TFRECORD
    idx1 = idx0 + IMAGE_PER_TFRECORD
    record_file = TFRECORD_PATH + '%05d.tfrecords' % idx
    with tf.io.TFRecordWriter(record_file) as writer:
        for filename, label in zip(file_list[idx0:idx1],
                                   label_list[idx0:idx1]):
            image_string = open(filename, 'rb').read()
            tf_example = image_example(image_string, label)
            writer.write(tf_example.SerializeToString())
```

Model learning

In [ ]:
```
! pip install tensorflow_addons
```

```
Collecting tensorflow_addons
  Downloading tensorflow_addons-0.14.0-cp37-cp37m-manylinux_2_12_x86_6
4.manylinux2010_x86_64.whl (1.1 MB)
     |████████████████████████████████| 1.1 MB 5.2 MB/s
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python
3.7/dist-packages (from tensorflow_addons) (2.7.1)
Installing collected packages: tensorflow-addons
Successfully installed tensorflow-addons-0.14.0
```

In [ ]:
```
import tensorflow_addons as tfa
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Conv2D, MaxPool2D, Concatenate, Flat
ten, Dense
```

Hyper parameter

In [ ]:
```
EPOCHS = 1000
RESULT_SAVE_PATH = 'results/'
```

Function define

Inception-based model function

In [ ]:
```
def Model():
    def inception(filters):
        def subnetwork(x):
            h1 = Conv2D(filters, (1, 1), padding='same', activation='rel
u')(x)
            h1 = MaxPool2D()(h1)

            h2 = Conv2D(filters // 2, (1, 1), padding='same', activation
='relu')(x)
            h2 = Conv2D(filters, (3, 3), padding='same', activation='rel
u')(h2)
            h2 = MaxPool2D()(h2)

            h3 = Conv2D(filters // 2, (1, 1), padding='same', activation
='relu')(x)
            h3 = Conv2D(filters, (5, 5), padding='same', activation='rel
u')(h3)
            h3 = MaxPool2D()(h3)
            return Concatenate()([h1, h2, h3])
        return subnetwork

    x = tf.keras.Input(shape=(256, 256, 3))
    h = inception(16)(x)
    h = inception(32)(h)
    h = inception(32)(h)
    h = inception(32)(h)
    h = inception(32)(h)
    h = Flatten()(h)
    h = Dense(1024, activation='relu')(h)
    y = Dense(1, activation='sigmoid')(h)
    return tf.keras.Model(inputs=x, outputs=y)
```

Data preprocessing function

In [ ]:
```
def preprocess(img):
    return tf.image.convert_image_dtype(img, tf.float32)
```

Data Augmentation function

- do filp, rotate, translation

In [ ]:
```
def augmentation(img, label):
    def flip(x):
        x = tf.image.random_flip_left_right(x)
        x = tf.image.random_flip_up_down(x)
        return x

    def rotate(x):
        x = tf.cond(tf.random.uniform(shape=[], minval=0.0, maxval=1.0,
dtype=tf.float32) > 0.5,
                    lambda: tfa.image.rotate(x,
                                    tf.random.uniform(shape=[], minva
l=0.0, maxval=360.0, dtype=tf.float32),
                                    interpolation='BILINEAR'),
                    lambda: x)
        return x

    def translation(x):
        dx = tf.random.uniform(shape=[], minval=-10.0, maxval=10.0, dtyp
e=tf.float32)
        dy = tf.random.uniform(shape=[], minval=-10.0, maxval=10.0, dtyp
e=tf.float32)
        x = tf.cond(tf.random.uniform(shape=[], minval=0.0, maxval=1.0,
dtype=tf.float32) > 0.5,
                     lambda: tfa.image.transform(x,
                                         [0, 0, dx, 0, 0, dy, 0,
0],
                                         interpolation='BILINEAR'
),
                     lambda: x)
        return x

    img = flip(img)
    img = rotate(img)
    img = translation(img)

    return img, label
```

Load TFRecords

In [ ]:
```
tffiles = glob.glob('tfrecords/*')
raw_image_dataset = tf.data.TFRecordDataset(tffiles)

image_feature_description = {
    'height': tf.io.FixedLenFeature([], tf.int64),
    'width': tf.io.FixedLenFeature([], tf.int64),
    'depth': tf.io.FixedLenFeature([], tf.int64),
    'label': tf.io.FixedLenFeature([], tf.int64),
    'image_raw': tf.io.FixedLenFeature([], tf.string),
}

def _parse_image_function(example_proto):
    return tf.io.parse_single_example(example_proto, image_feature_descr
iption)

def _parse_image_label(parsed_dataset):
    return preprocess(tf.image.decode_png(parsed_dataset['image_raw'])),
parsed_dataset['label']

parsed_image_dataset = raw_image_dataset.map(_parse_image_function)
dataset = parsed_image_dataset.map(_parse_image_label)
```

Train and Validation set

In [ ]:
```
ds_size = 0
for _ in dataset:
    ds_size += 1

train_size = int(ds_size * 0.7)

ds = dataset.shuffle(ds_size)
ds_train = ds.take(train_size).shuffle(1024, reshuffle_each_iteration=Tr
ue).prefetch(1024).batch(32).map(augmentation)
ds_valid = ds.skip(train_size).prefetch(1024).batch(32)
```

Build a model and start learning

In [ ]:
```
model = Model()
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

In [ ]:
```
earlystopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', pat
ience=20, verbose=1)
history = model.fit(ds_train,
                    validation_data=ds_valid,
                    epochs=EPOCHS,
                    callbacks=[earlystopping])
```

```
Epoch 1/1000
58/58 [==============================] - 70s 547ms/step - loss: 0.6938
- accuracy: 0.5261 - val_loss: 0.6901 - val_accuracy: 0.4937
Epoch 2/1000
58/58 [==============================] - 34s 522ms/step - loss: 0.6935
- accuracy: 0.5087 - val_loss: 0.6919 - val_accuracy: 0.5076
Epoch 3/1000
58/58 [==============================] - 33s 502ms/step - loss: 0.6930
- accuracy: 0.4995 - val_loss: 0.6874 - val_accuracy: 0.4987
Epoch 4/1000
58/58 [==============================] - 34s 504ms/step - loss: 0.6845
- accuracy: 0.5506 - val_loss: 0.6865 - val_accuracy: 0.5723
Epoch 5/1000
58/58 [==============================] - 34s 514ms/step - loss: 0.6871
- accuracy: 0.5256 - val_loss: 0.6854 - val_accuracy: 0.5063
Epoch 6/1000
58/58 [==============================] - 34s 504ms/step - loss: 0.6937
- accuracy: 0.5147 - val_loss: 0.6887 - val_accuracy: 0.6409
Epoch 7/1000
58/58 [==============================] - 33s 502ms/step - loss: 0.6912
- accuracy: 0.5234 - val_loss: 0.6879 - val_accuracy: 0.5203
Epoch 8/1000
58/58 [==============================] - 34s 510ms/step - loss: 0.6889
- accuracy: 0.5370 - val_loss: 0.7002 - val_accuracy: 0.5025
Epoch 9/1000
58/58 [==============================] - 33s 498ms/step - loss: 0.6883
- accuracy: 0.5397 - val_loss: 1.0727 - val_accuracy: 0.5165
Epoch 10/1000
58/58 [==============================] - 34s 512ms/step - loss: 0.7020
- accuracy: 0.5109 - val_loss: 0.6948 - val_accuracy: 0.4480
Epoch 11/1000
58/58 [==============================] - 33s 490ms/step - loss: 0.6812
- accuracy: 0.5354 - val_loss: 0.6628 - val_accuracy: 0.6548
Epoch 12/1000
58/58 [==============================] - 35s 520ms/step - loss: 0.6881
- accuracy: 0.5109 - val_loss: 0.6914 - val_accuracy: 0.6053
Epoch 13/1000
58/58 [==============================] - 33s 495ms/step - loss: 0.6885
- accuracy: 0.5147 - val_loss: 0.6783 - val_accuracy: 0.5888
Epoch 14/1000
58/58 [==============================] - 34s 506ms/step - loss: 0.6815
- accuracy: 0.5664 - val_loss: 0.6555 - val_accuracy: 0.5787
Epoch 15/1000
58/58 [==============================] - 33s 507ms/step - loss: 0.6650
- accuracy: 0.5881 - val_loss: 0.6438 - val_accuracy: 0.6612
Epoch 16/1000
58/58 [==============================] - 32s 484ms/step - loss: 0.6824
- accuracy: 0.5375 - val_loss: 0.6582 - val_accuracy: 0.6586
Epoch 17/1000
58/58 [==============================] - 34s 510ms/step - loss: 0.6618
- accuracy: 0.6001 - val_loss: 0.6141 - val_accuracy: 0.6853
Epoch 18/1000
58/58 [==============================] - 33s 496ms/step - loss: 0.6496
- accuracy: 0.5996 - val_loss: 0.6260 - val_accuracy: 0.6675
Epoch 19/1000
58/58 [==============================] - 34s 514ms/step - loss: 0.6530
- accuracy: 0.5930 - val_loss: 0.6628 - val_accuracy: 0.6091
```

```
Epoch 20/1000
58/58 [==============================] - 33s 490ms/step - loss: 0.6640
- accuracy: 0.5484 - val_loss: 0.6167 - val_accuracy: 0.6764
Epoch 21/1000
58/58 [==============================] - 33s 496ms/step - loss: 0.6433
- accuracy: 0.5985 - val_loss: 0.6411 - val_accuracy: 0.6853
Epoch 22/1000
58/58 [==============================] - 33s 497ms/step - loss: 0.6492
- accuracy: 0.6045 - val_loss: 0.6002 - val_accuracy: 0.6574
Epoch 23/1000
58/58 [==============================] - 33s 491ms/step - loss: 0.6449
- accuracy: 0.6088 - val_loss: 0.6154 - val_accuracy: 0.6117
Epoch 24/1000
58/58 [==============================] - 33s 493ms/step - loss: 0.6581
- accuracy: 0.5615 - val_loss: 0.5975 - val_accuracy: 0.6561
Epoch 25/1000
58/58 [==============================] - 32s 482ms/step - loss: 0.6358
- accuracy: 0.6115 - val_loss: 0.5642 - val_accuracy: 0.7195
Epoch 26/1000
58/58 [==============================] - 34s 505ms/step - loss: 0.6318
- accuracy: 0.5990 - val_loss: 0.5437 - val_accuracy: 0.7538
Epoch 27/1000
58/58 [==============================] - 34s 519ms/step - loss: 0.6509
- accuracy: 0.5854 - val_loss: 0.5922 - val_accuracy: 0.7069
Epoch 28/1000
58/58 [==============================] - 33s 494ms/step - loss: 0.6338
- accuracy: 0.5996 - val_loss: 0.5952 - val_accuracy: 0.7398
Epoch 29/1000
58/58 [==============================] - 33s 500ms/step - loss: 0.6463
- accuracy: 0.5871 - val_loss: 0.5400 - val_accuracy: 0.7360
Epoch 30/1000
58/58 [==============================] - 33s 499ms/step - loss: 0.6497
- accuracy: 0.5892 - val_loss: 0.6298 - val_accuracy: 0.6802
Epoch 31/1000
58/58 [==============================] - 32s 479ms/step - loss: 0.6527
- accuracy: 0.5773 - val_loss: 0.5538 - val_accuracy: 0.7310
Epoch 32/1000
58/58 [==============================] - 33s 488ms/step - loss: 0.6402
- accuracy: 0.6186 - val_loss: 0.5983 - val_accuracy: 0.6764
Epoch 33/1000
58/58 [==============================] - 32s 487ms/step - loss: 0.6298
- accuracy: 0.6072 - val_loss: 0.5571 - val_accuracy: 0.7272
Epoch 34/1000
58/58 [==============================] - 33s 490ms/step - loss: 0.6230
- accuracy: 0.6137 - val_loss: 0.5717 - val_accuracy: 0.7018
Epoch 35/1000
58/58 [==============================] - 32s 484ms/step - loss: 0.6242
- accuracy: 0.6300 - val_loss: 0.5357 - val_accuracy: 0.7538
Epoch 36/1000
58/58 [==============================] - 33s 502ms/step - loss: 0.6327
- accuracy: 0.5930 - val_loss: 0.5702 - val_accuracy: 0.7576
Epoch 37/1000
58/58 [==============================] - 32s 487ms/step - loss: 0.6259
- accuracy: 0.6425 - val_loss: 0.5498 - val_accuracy: 0.7589
Epoch 38/1000
58/58 [==============================] - 33s 490ms/step - loss: 0.6180
- accuracy: 0.6496 - val_loss: 0.5431 - val_accuracy: 0.7360
```

```
Epoch 39/1000
58/58 [==============================] - 34s 509ms/step - loss: 0.6258
- accuracy: 0.6213 - val_loss: 0.5213 - val_accuracy: 0.7589
Epoch 40/1000
58/58 [==============================] - 33s 503ms/step - loss: 0.6304
- accuracy: 0.6317 - val_loss: 0.5640 - val_accuracy: 0.7551
Epoch 41/1000
58/58 [==============================] - 34s 511ms/step - loss: 0.6268
- accuracy: 0.6126 - val_loss: 0.5465 - val_accuracy: 0.7437
Epoch 42/1000
58/58 [==============================] - 34s 510ms/step - loss: 0.6189
- accuracy: 0.6387 - val_loss: 0.5703 - val_accuracy: 0.7069
Epoch 43/1000
58/58 [==============================] - 34s 504ms/step - loss: 0.6209
- accuracy: 0.6235 - val_loss: 0.5368 - val_accuracy: 0.7145
Epoch 44/1000
58/58 [==============================] - 34s 508ms/step - loss: 0.6318
- accuracy: 0.6170 - val_loss: 0.5874 - val_accuracy: 0.7538
Epoch 45/1000
58/58 [==============================] - 32s 488ms/step - loss: 0.6390
- accuracy: 0.6159 - val_loss: 0.5402 - val_accuracy: 0.7500
Epoch 46/1000
58/58 [==============================] - 35s 524ms/step - loss: 0.6638
- accuracy: 0.6153 - val_loss: 0.5381 - val_accuracy: 0.7310
Epoch 47/1000
58/58 [==============================] - 34s 506ms/step - loss: 0.6185
- accuracy: 0.6224 - val_loss: 0.5884 - val_accuracy: 0.6409
Epoch 48/1000
58/58 [==============================] - 33s 494ms/step - loss: 0.6349
- accuracy: 0.6197 - val_loss: 0.5039 - val_accuracy: 0.7576
Epoch 49/1000
58/58 [==============================] - 33s 497ms/step - loss: 0.6231
- accuracy: 0.6262 - val_loss: 0.5116 - val_accuracy: 0.7741
Epoch 50/1000
58/58 [==============================] - 33s 501ms/step - loss: 0.6338
- accuracy: 0.5985 - val_loss: 0.5063 - val_accuracy: 0.7741
Epoch 51/1000
58/58 [==============================] - 32s 485ms/step - loss: 0.6066
- accuracy: 0.6208 - val_loss: 0.5087 - val_accuracy: 0.7652
Epoch 52/1000
58/58 [==============================] - 33s 489ms/step - loss: 0.5843
- accuracy: 0.6665 - val_loss: 0.4794 - val_accuracy: 0.7690
Epoch 53/1000
58/58 [==============================] - 32s 476ms/step - loss: 0.5956
- accuracy: 0.6496 - val_loss: 0.5256 - val_accuracy: 0.7627
Epoch 54/1000
58/58 [==============================] - 32s 477ms/step - loss: 0.5691
- accuracy: 0.6888 - val_loss: 0.4940 - val_accuracy: 0.7779
Epoch 55/1000
58/58 [==============================] - 33s 490ms/step - loss: 0.6355
- accuracy: 0.6257 - val_loss: 0.6095 - val_accuracy: 0.6853
Epoch 56/1000
58/58 [==============================] - 32s 484ms/step - loss: 0.6263
- accuracy: 0.6376 - val_loss: 0.5176 - val_accuracy: 0.7690
Epoch 57/1000
58/58 [==============================] - 32s 490ms/step - loss: 0.6192
- accuracy: 0.6279 - val_loss: 0.5230 - val_accuracy: 0.7221
```

```
Epoch 58/1000
58/58 [==============================] - 34s 505ms/step - loss: 0.5955
- accuracy: 0.6485 - val_loss: 0.5118 - val_accuracy: 0.7589
Epoch 59/1000
58/58 [==============================] - 34s 508ms/step - loss: 0.6097
- accuracy: 0.6333 - val_loss: 0.5106 - val_accuracy: 0.7538
Epoch 60/1000
58/58 [==============================] - 34s 517ms/step - loss: 0.6095
- accuracy: 0.6322 - val_loss: 0.5115 - val_accuracy: 0.7525
Epoch 61/1000
58/58 [==============================] - 33s 496ms/step - loss: 0.5954
- accuracy: 0.6311 - val_loss: 0.4538 - val_accuracy: 0.7766
Epoch 62/1000
58/58 [==============================] - 32s 474ms/step - loss: 0.5901
- accuracy: 0.6453 - val_loss: 0.4784 - val_accuracy: 0.7766
Epoch 63/1000
58/58 [==============================] - 34s 521ms/step - loss: 0.6023
- accuracy: 0.6251 - val_loss: 0.4884 - val_accuracy: 0.7652
Epoch 64/1000
58/58 [==============================] - 34s 504ms/step - loss: 0.6084
- accuracy: 0.6066 - val_loss: 0.5105 - val_accuracy: 0.7805
Epoch 65/1000
58/58 [==============================] - 32s 483ms/step - loss: 0.6003
- accuracy: 0.6366 - val_loss: 0.4739 - val_accuracy: 0.7817
Epoch 66/1000
58/58 [==============================] - 33s 499ms/step - loss: 0.6082
- accuracy: 0.6322 - val_loss: 0.4762 - val_accuracy: 0.7703
Epoch 67/1000
58/58 [==============================] - 34s 506ms/step - loss: 0.5947
- accuracy: 0.6425 - val_loss: 0.4718 - val_accuracy: 0.7703
Epoch 68/1000
58/58 [==============================] - 34s 510ms/step - loss: 0.5848
- accuracy: 0.6376 - val_loss: 0.4296 - val_accuracy: 0.7893
Epoch 69/1000
58/58 [==============================] - 33s 499ms/step - loss: 0.5887
- accuracy: 0.6480 - val_loss: 0.4576 - val_accuracy: 0.7728
Epoch 70/1000
58/58 [==============================] - 33s 507ms/step - loss: 0.5943
- accuracy: 0.6240 - val_loss: 0.4630 - val_accuracy: 0.7703
Epoch 71/1000
58/58 [==============================] - 32s 488ms/step - loss: 0.5668
- accuracy: 0.6578 - val_loss: 0.4467 - val_accuracy: 0.7690
Epoch 72/1000
58/58 [==============================] - 33s 499ms/step - loss: 0.5836
- accuracy: 0.6523 - val_loss: 0.4786 - val_accuracy: 0.7627
Epoch 73/1000
58/58 [==============================] - 33s 498ms/step - loss: 0.5980
- accuracy: 0.6017 - val_loss: 0.4148 - val_accuracy: 0.8020
Epoch 74/1000
58/58 [==============================] - 34s 518ms/step - loss: 0.5859
- accuracy: 0.6398 - val_loss: 0.4549 - val_accuracy: 0.7665
Epoch 75/1000
58/58 [==============================] - 33s 504ms/step - loss: 0.5701
- accuracy: 0.6376 - val_loss: 0.4251 - val_accuracy: 0.7678
Epoch 76/1000
58/58 [==============================] - 33s 505ms/step - loss: 0.5744
- accuracy: 0.6360 - val_loss: 0.5738 - val_accuracy: 0.7602
```

```
Epoch 77/1000
58/58 [==============================] - 33s 488ms/step - loss: 0.5425
- accuracy: 0.6730 - val_loss: 0.5869 - val_accuracy: 0.7424
Epoch 78/1000
58/58 [==============================] - 32s 486ms/step - loss: 0.5890
- accuracy: 0.6300 - val_loss: 0.4779 - val_accuracy: 0.7779
Epoch 79/1000
58/58 [==============================] - 33s 494ms/step - loss: 0.5854
- accuracy: 0.6360 - val_loss: 0.4214 - val_accuracy: 0.7690
Epoch 80/1000
58/58 [==============================] - 33s 493ms/step - loss: 0.5755
- accuracy: 0.6474 - val_loss: 0.4046 - val_accuracy: 0.7944
Epoch 81/1000
58/58 [==============================] - 33s 492ms/step - loss: 0.6155
- accuracy: 0.6001 - val_loss: 0.5451 - val_accuracy: 0.6992
Epoch 82/1000
58/58 [==============================] - 34s 514ms/step - loss: 0.6493
- accuracy: 0.5550 - val_loss: 0.5187 - val_accuracy: 0.7310
Epoch 83/1000
58/58 [==============================] - 33s 494ms/step - loss: 0.6006
- accuracy: 0.6398 - val_loss: 0.4725 - val_accuracy: 0.7665
Epoch 84/1000
58/58 [==============================] - 33s 488ms/step - loss: 0.5474
- accuracy: 0.6741 - val_loss: 0.4499 - val_accuracy: 0.7982
Epoch 85/1000
58/58 [==============================] - 33s 501ms/step - loss: 0.5868
- accuracy: 0.6208 - val_loss: 0.4213 - val_accuracy: 0.7805
Epoch 86/1000
58/58 [==============================] - 33s 495ms/step - loss: 0.5858
- accuracy: 0.6338 - val_loss: 0.4892 - val_accuracy: 0.7525
Epoch 87/1000
58/58 [==============================] - 34s 509ms/step - loss: 0.5641
- accuracy: 0.6502 - val_loss: 0.4112 - val_accuracy: 0.7855
Epoch 88/1000
58/58 [==============================] - 36s 540ms/step - loss: 0.5950
- accuracy: 0.6126 - val_loss: 0.4618 - val_accuracy: 0.7995
Epoch 89/1000
58/58 [==============================] - 34s 512ms/step - loss: 0.5836
- accuracy: 0.6300 - val_loss: 0.4284 - val_accuracy: 0.8147
Epoch 90/1000
58/58 [==============================] - 33s 491ms/step - loss: 0.5717
- accuracy: 0.6523 - val_loss: 0.5070 - val_accuracy: 0.7652
Epoch 91/1000
58/58 [==============================] - 33s 504ms/step - loss: 0.6156
- accuracy: 0.5979 - val_loss: 0.4417 - val_accuracy: 0.7919
Epoch 92/1000
58/58 [==============================] - 33s 503ms/step - loss: 0.5805
- accuracy: 0.6186 - val_loss: 0.4389 - val_accuracy: 0.7766
Epoch 93/1000
58/58 [==============================] - 33s 487ms/step - loss: 0.5416
- accuracy: 0.6632 - val_loss: 0.4269 - val_accuracy: 0.8046
Epoch 94/1000
58/58 [==============================] - 34s 507ms/step - loss: 0.5977
- accuracy: 0.6344 - val_loss: 0.4651 - val_accuracy: 0.7830
Epoch 95/1000
58/58 [==============================] - 34s 513ms/step - loss: 0.5813
- accuracy: 0.6159 - val_loss: 0.3928 - val_accuracy: 0.7970
```

```
Epoch 96/1000
58/58 [==============================] - 33s 491ms/step - loss: 0.5483
- accuracy: 0.6692 - val_loss: 0.4032 - val_accuracy: 0.8008
Epoch 97/1000
58/58 [==============================] - 32s 479ms/step - loss: 0.5242
- accuracy: 0.6855 - val_loss: 0.3887 - val_accuracy: 0.8096
Epoch 98/1000
58/58 [==============================] - 33s 498ms/step - loss: 0.5743
- accuracy: 0.6300 - val_loss: 0.3953 - val_accuracy: 0.8363
Epoch 99/1000
58/58 [==============================] - 32s 478ms/step - loss: 0.5674
- accuracy: 0.6605 - val_loss: 0.3729 - val_accuracy: 0.8185
Epoch 100/1000
58/58 [==============================] - 34s 518ms/step - loss: 0.5909
- accuracy: 0.6322 - val_loss: 0.3811 - val_accuracy: 0.8338
Epoch 101/1000
58/58 [==============================] - 33s 492ms/step - loss: 0.5502
- accuracy: 0.6513 - val_loss: 0.3619 - val_accuracy: 0.8274
Epoch 102/1000
58/58 [==============================] - 33s 491ms/step - loss: 0.5564
- accuracy: 0.6556 - val_loss: 0.3546 - val_accuracy: 0.8439
Epoch 103/1000
58/58 [==============================] - 34s 514ms/step - loss: 0.5469
- accuracy: 0.6474 - val_loss: 0.4209 - val_accuracy: 0.7817
Epoch 104/1000
58/58 [==============================] - 33s 504ms/step - loss: 0.5592
- accuracy: 0.6502 - val_loss: 0.3833 - val_accuracy: 0.8147
Epoch 105/1000
58/58 [==============================] - 34s 514ms/step - loss: 0.5772
- accuracy: 0.6148 - val_loss: 0.3661 - val_accuracy: 0.8135
Epoch 106/1000
58/58 [==============================] - 33s 491ms/step - loss: 0.5462
- accuracy: 0.6556 - val_loss: 0.3621 - val_accuracy: 0.8363
Epoch 107/1000
58/58 [==============================] - 34s 506ms/step - loss: 0.5450
- accuracy: 0.6621 - val_loss: 0.4150 - val_accuracy: 0.7995
Epoch 108/1000
58/58 [==============================] - 34s 503ms/step - loss: 0.6153
- accuracy: 0.6012 - val_loss: 0.4540 - val_accuracy: 0.7931
Epoch 109/1000
58/58 [==============================] - 34s 506ms/step - loss: 0.5774
- accuracy: 0.6507 - val_loss: 0.4562 - val_accuracy: 0.7855
Epoch 110/1000
58/58 [==============================] - 34s 512ms/step - loss: 0.5502
- accuracy: 0.6583 - val_loss: 0.3977 - val_accuracy: 0.8198
Epoch 111/1000
58/58 [==============================] - 34s 508ms/step - loss: 0.5333
- accuracy: 0.6643 - val_loss: 0.3679 - val_accuracy: 0.8579
Epoch 112/1000
58/58 [==============================] - 34s 518ms/step - loss: 0.5673
- accuracy: 0.6556 - val_loss: 0.4779 - val_accuracy: 0.7259
Epoch 113/1000
58/58 [==============================] - 32s 471ms/step - loss: 0.5451
- accuracy: 0.6697 - val_loss: 0.4523 - val_accuracy: 0.7779
Epoch 114/1000
58/58 [==============================] - 34s 515ms/step - loss: 0.5673
- accuracy: 0.6507 - val_loss: 0.3655 - val_accuracy: 0.8693
```

```
Epoch 115/1000
58/58 [==============================] - 33s 497ms/step - loss: 0.5623
- accuracy: 0.6262 - val_loss: 0.3844 - val_accuracy: 0.8464
Epoch 116/1000
58/58 [==============================] - 35s 522ms/step - loss: 0.5651
- accuracy: 0.6279 - val_loss: 0.3203 - val_accuracy: 0.8642
Epoch 117/1000
58/58 [==============================] - 34s 517ms/step - loss: 0.5607
- accuracy: 0.6382 - val_loss: 0.4405 - val_accuracy: 0.7931
Epoch 118/1000
58/58 [==============================] - 34s 518ms/step - loss: 0.6065
- accuracy: 0.6251 - val_loss: 0.4438 - val_accuracy: 0.7995
Epoch 119/1000
58/58 [==============================] - 34s 505ms/step - loss: 0.5646
- accuracy: 0.6621 - val_loss: 0.4249 - val_accuracy: 0.8198
Epoch 120/1000
58/58 [==============================] - 33s 502ms/step - loss: 0.5301
- accuracy: 0.6801 - val_loss: 0.4560 - val_accuracy: 0.8249
Epoch 121/1000
58/58 [==============================] - 33s 505ms/step - loss: 0.5480
- accuracy: 0.6638 - val_loss: 0.3211 - val_accuracy: 0.8668
Epoch 122/1000
58/58 [==============================] - 33s 498ms/step - loss: 0.5310
- accuracy: 0.6649 - val_loss: 0.4186 - val_accuracy: 0.8096
Epoch 123/1000
58/58 [==============================] - 32s 485ms/step - loss: 0.5484
- accuracy: 0.6741 - val_loss: 0.3165 - val_accuracy: 0.8744
Epoch 124/1000
58/58 [==============================] - 33s 498ms/step - loss: 0.5194
- accuracy: 0.6730 - val_loss: 0.3165 - val_accuracy: 0.8629
Epoch 125/1000
58/58 [==============================] - 32s 476ms/step - loss: 0.5163
- accuracy: 0.6942 - val_loss: 0.3375 - val_accuracy: 0.8579
Epoch 126/1000
58/58 [==============================] - 33s 499ms/step - loss: 0.5275
- accuracy: 0.6746 - val_loss: 0.3942 - val_accuracy: 0.8541
Epoch 127/1000
58/58 [==============================] - 34s 516ms/step - loss: 0.5212
- accuracy: 0.6507 - val_loss: 0.2902 - val_accuracy: 0.8832
Epoch 128/1000
58/58 [==============================] - 33s 502ms/step - loss: 0.5034
- accuracy: 0.6872 - val_loss: 0.2487 - val_accuracy: 0.9010
Epoch 129/1000
58/58 [==============================] - 34s 511ms/step - loss: 0.5234
- accuracy: 0.6980 - val_loss: 0.5745 - val_accuracy: 0.7462
Epoch 130/1000
58/58 [==============================] - 34s 516ms/step - loss: 0.5767
- accuracy: 0.6436 - val_loss: 0.3549 - val_accuracy: 0.8820
Epoch 131/1000
58/58 [==============================] - 34s 519ms/step - loss: 0.4976
- accuracy: 0.6888 - val_loss: 0.3101 - val_accuracy: 0.8896
Epoch 132/1000
58/58 [==============================] - 34s 507ms/step - loss: 0.4798
- accuracy: 0.7209 - val_loss: 0.2674 - val_accuracy: 0.8985
Epoch 133/1000
58/58 [==============================] - 33s 499ms/step - loss: 0.5121
- accuracy: 0.6850 - val_loss: 0.2525 - val_accuracy: 0.8959
```

```
Epoch 134/1000
58/58 [==============================] - 34s 517ms/step - loss: 0.5116
- accuracy: 0.6980 - val_loss: 0.2803 - val_accuracy: 0.9036
Epoch 135/1000
58/58 [==============================] - 32s 484ms/step - loss: 0.5248
- accuracy: 0.6714 - val_loss: 0.2065 - val_accuracy: 0.9264
Epoch 136/1000
58/58 [==============================] - 34s 519ms/step - loss: 0.5224
- accuracy: 0.6741 - val_loss: 0.3632 - val_accuracy: 0.8604
Epoch 137/1000
58/58 [==============================] - 34s 521ms/step - loss: 0.4820
- accuracy: 0.7133 - val_loss: 0.2343 - val_accuracy: 0.9099
Epoch 138/1000
58/58 [==============================] - 33s 497ms/step - loss: 0.4647
- accuracy: 0.7160 - val_loss: 0.2446 - val_accuracy: 0.9099
Epoch 139/1000
58/58 [==============================] - 34s 509ms/step - loss: 0.5100
- accuracy: 0.6774 - val_loss: 0.2152 - val_accuracy: 0.9264
Epoch 140/1000
58/58 [==============================] - 34s 509ms/step - loss: 0.4876
- accuracy: 0.6948 - val_loss: 0.2240 - val_accuracy: 0.9124
Epoch 141/1000
58/58 [==============================] - 35s 521ms/step - loss: 0.5208
- accuracy: 0.6736 - val_loss: 0.2751 - val_accuracy: 0.8959
Epoch 142/1000
58/58 [==============================] - 33s 493ms/step - loss: 0.5215
- accuracy: 0.6763 - val_loss: 0.1861 - val_accuracy: 0.9353
Epoch 143/1000
58/58 [==============================] - 33s 492ms/step - loss: 0.4573
- accuracy: 0.7193 - val_loss: 0.2903 - val_accuracy: 0.8794
Epoch 144/1000
58/58 [==============================] - 34s 510ms/step - loss: 0.4711
- accuracy: 0.7160 - val_loss: 0.2530 - val_accuracy: 0.9061
Epoch 145/1000
58/58 [==============================] - 34s 518ms/step - loss: 0.5067
- accuracy: 0.6828 - val_loss: 0.2352 - val_accuracy: 0.8997
Epoch 146/1000
58/58 [==============================] - 33s 496ms/step - loss: 0.4959
- accuracy: 0.6921 - val_loss: 0.2376 - val_accuracy: 0.9137
Epoch 147/1000
58/58 [==============================] - 32s 473ms/step - loss: 0.4391
- accuracy: 0.7339 - val_loss: 0.2309 - val_accuracy: 0.9150
Epoch 148/1000
58/58 [==============================] - 33s 489ms/step - loss: 0.5134
- accuracy: 0.6768 - val_loss: 0.2459 - val_accuracy: 0.9099
Epoch 149/1000
58/58 [==============================] - 35s 531ms/step - loss: 0.5112
- accuracy: 0.6790 - val_loss: 0.2252 - val_accuracy: 0.9086
Epoch 150/1000
58/58 [==============================] - 32s 486ms/step - loss: 0.4445
- accuracy: 0.7356 - val_loss: 0.2276 - val_accuracy: 0.9239
Epoch 151/1000
58/58 [==============================] - 33s 501ms/step - loss: 0.4611
- accuracy: 0.7193 - val_loss: 0.2083 - val_accuracy: 0.9277
Epoch 152/1000
58/58 [==============================] - 34s 514ms/step - loss: 0.4963
- accuracy: 0.7013 - val_loss: 0.2099 - val_accuracy: 0.9074
```

```
Epoch 153/1000
58/58 [==============================] - 33s 504ms/step - loss: 0.4584
- accuracy: 0.7149 - val_loss: 0.1910 - val_accuracy: 0.9213
Epoch 154/1000
58/58 [==============================] - 33s 505ms/step - loss: 0.4365
- accuracy: 0.7394 - val_loss: 0.1975 - val_accuracy: 0.9162
Epoch 155/1000
58/58 [==============================] - 33s 489ms/step - loss: 0.4197
- accuracy: 0.7361 - val_loss: 0.2042 - val_accuracy: 0.9226
Epoch 156/1000
58/58 [==============================] - 34s 511ms/step - loss: 0.4574
- accuracy: 0.7274 - val_loss: 0.2051 - val_accuracy: 0.9226
Epoch 157/1000
58/58 [==============================] - 34s 507ms/step - loss: 0.4566
- accuracy: 0.7182 - val_loss: 0.1980 - val_accuracy: 0.9188
Epoch 158/1000
58/58 [==============================] - 32s 480ms/step - loss: 0.4276
- accuracy: 0.7361 - val_loss: 0.1838 - val_accuracy: 0.9315
Epoch 159/1000
58/58 [==============================] - 33s 496ms/step - loss: 0.4512
- accuracy: 0.7301 - val_loss: 0.1947 - val_accuracy: 0.9289
Epoch 160/1000
58/58 [==============================] - 33s 495ms/step - loss: 0.4550
- accuracy: 0.7116 - val_loss: 0.2224 - val_accuracy: 0.9099
Epoch 161/1000
58/58 [==============================] - 33s 496ms/step - loss: 0.4377
- accuracy: 0.7291 - val_loss: 0.1968 - val_accuracy: 0.9239
Epoch 162/1000
58/58 [==============================] - 34s 508ms/step - loss: 0.4837
- accuracy: 0.6980 - val_loss: 0.2047 - val_accuracy: 0.9201
Epoch 163/1000
58/58 [==============================] - 34s 513ms/step - loss: 0.4286
- accuracy: 0.7291 - val_loss: 0.1625 - val_accuracy: 0.9365
Epoch 164/1000
58/58 [==============================] - 34s 520ms/step - loss: 0.4779
- accuracy: 0.6986 - val_loss: 0.1881 - val_accuracy: 0.9340
Epoch 165/1000
58/58 [==============================] - 34s 506ms/step - loss: 0.4768
- accuracy: 0.7111 - val_loss: 0.1488 - val_accuracy: 0.9530
Epoch 166/1000
58/58 [==============================] - 34s 510ms/step - loss: 0.4412
- accuracy: 0.7144 - val_loss: 0.2037 - val_accuracy: 0.9188
Epoch 167/1000
58/58 [==============================] - 35s 525ms/step - loss: 0.5103
- accuracy: 0.6882 - val_loss: 0.1701 - val_accuracy: 0.9327
Epoch 168/1000
58/58 [==============================] - 34s 518ms/step - loss: 0.4813
- accuracy: 0.6910 - val_loss: 0.2170 - val_accuracy: 0.9112
Epoch 169/1000
58/58 [==============================] - 34s 505ms/step - loss: 0.4690
- accuracy: 0.7013 - val_loss: 0.1597 - val_accuracy: 0.9365
Epoch 170/1000
58/58 [==============================] - 34s 508ms/step - loss: 0.4264
- accuracy: 0.7345 - val_loss: 0.1940 - val_accuracy: 0.9289
Epoch 171/1000
58/58 [==============================] - 32s 474ms/step - loss: 0.3941
- accuracy: 0.7519 - val_loss: 0.1960 - val_accuracy: 0.9226
```

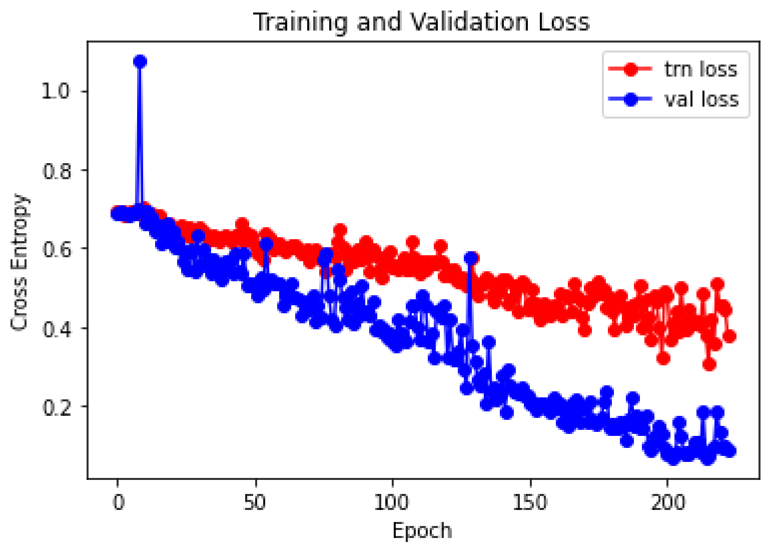
```
Epoch 172/1000
58/58 [==============================] - 34s 512ms/step - loss: 0.4603
- accuracy: 0.7073 - val_loss: 0.1582 - val_accuracy: 0.9442
Epoch 173/1000
58/58 [==============================] - 34s 511ms/step - loss: 0.5019
- accuracy: 0.6719 - val_loss: 0.2089 - val_accuracy: 0.9150
Epoch 174/1000
58/58 [==============================] - 34s 507ms/step - loss: 0.4760
- accuracy: 0.6964 - val_loss: 0.1662 - val_accuracy: 0.9365
Epoch 175/1000
58/58 [==============================] - 34s 521ms/step - loss: 0.4855
- accuracy: 0.6774 - val_loss: 0.1573 - val_accuracy: 0.9365
Epoch 176/1000
58/58 [==============================] - 34s 522ms/step - loss: 0.5159
- accuracy: 0.6491 - val_loss: 0.1599 - val_accuracy: 0.9277
Epoch 177/1000
58/58 [==============================] - 34s 515ms/step - loss: 0.4855
- accuracy: 0.6801 - val_loss: 0.1687 - val_accuracy: 0.9226
Epoch 178/1000
58/58 [==============================] - 33s 495ms/step - loss: 0.4927
- accuracy: 0.6730 - val_loss: 0.2105 - val_accuracy: 0.9251
Epoch 179/1000
58/58 [==============================] - 34s 515ms/step - loss: 0.4485
- accuracy: 0.7203 - val_loss: 0.2380 - val_accuracy: 0.9188
Epoch 180/1000
58/58 [==============================] - 34s 509ms/step - loss: 0.4757
- accuracy: 0.6828 - val_loss: 0.1451 - val_accuracy: 0.9365
Epoch 181/1000
58/58 [==============================] - 34s 511ms/step - loss: 0.4412
- accuracy: 0.7122 - val_loss: 0.1556 - val_accuracy: 0.9378
Epoch 182/1000
58/58 [==============================] - 33s 488ms/step - loss: 0.3962
- accuracy: 0.7476 - val_loss: 0.1471 - val_accuracy: 0.9277
Epoch 183/1000
58/58 [==============================] - 34s 519ms/step - loss: 0.4665
- accuracy: 0.6882 - val_loss: 0.1448 - val_accuracy: 0.9327
Epoch 184/1000
58/58 [==============================] - 34s 503ms/step - loss: 0.4789
- accuracy: 0.6752 - val_loss: 0.1620 - val_accuracy: 0.9429
Epoch 185/1000
58/58 [==============================] - 33s 501ms/step - loss: 0.4500
- accuracy: 0.7209 - val_loss: 0.1480 - val_accuracy: 0.9327
Epoch 186/1000
58/58 [==============================] - 34s 517ms/step - loss: 0.4049
- accuracy: 0.7470 - val_loss: 0.1133 - val_accuracy: 0.9683
Epoch 187/1000
58/58 [==============================] - 33s 490ms/step - loss: 0.4261
- accuracy: 0.6986 - val_loss: 0.1649 - val_accuracy: 0.9353
Epoch 188/1000
58/58 [==============================] - 33s 493ms/step - loss: 0.4535
- accuracy: 0.7334 - val_loss: 0.2222 - val_accuracy: 0.9112
Epoch 189/1000
58/58 [==============================] - 34s 514ms/step - loss: 0.4292
- accuracy: 0.7296 - val_loss: 0.1522 - val_accuracy: 0.9429
Epoch 190/1000
58/58 [==============================] - 33s 501ms/step - loss: 0.4600
- accuracy: 0.7018 - val_loss: 0.1763 - val_accuracy: 0.9175
```

```
Epoch 191/1000
58/58 [==============================] - 35s 526ms/step - loss: 0.5062
- accuracy: 0.6556 - val_loss: 0.1464 - val_accuracy: 0.9353
Epoch 192/1000
58/58 [==============================] - 33s 492ms/step - loss: 0.4000
- accuracy: 0.7361 - val_loss: 0.1456 - val_accuracy: 0.9302
Epoch 193/1000
58/58 [==============================] - 34s 519ms/step - loss: 0.4716
- accuracy: 0.7040 - val_loss: 0.1769 - val_accuracy: 0.9277
Epoch 194/1000
58/58 [==============================] - 34s 504ms/step - loss: 0.4226
- accuracy: 0.7116 - val_loss: 0.0975 - val_accuracy: 0.9632
Epoch 195/1000
58/58 [==============================] - 33s 501ms/step - loss: 0.3682
- accuracy: 0.7655 - val_loss: 0.0908 - val_accuracy: 0.9683
Epoch 196/1000
58/58 [==============================] - 35s 524ms/step - loss: 0.4775
- accuracy: 0.6687 - val_loss: 0.1230 - val_accuracy: 0.9543
Epoch 197/1000
58/58 [==============================] - 33s 502ms/step - loss: 0.4637
- accuracy: 0.6964 - val_loss: 0.1144 - val_accuracy: 0.9657
Epoch 198/1000
58/58 [==============================] - 32s 483ms/step - loss: 0.4003
- accuracy: 0.7448 - val_loss: 0.1504 - val_accuracy: 0.9492
Epoch 199/1000
58/58 [==============================] - 33s 489ms/step - loss: 0.3251
- accuracy: 0.7922 - val_loss: 0.1286 - val_accuracy: 0.9632
Epoch 200/1000
58/58 [==============================] - 33s 500ms/step - loss: 0.4912
- accuracy: 0.6850 - val_loss: 0.0936 - val_accuracy: 0.9759
Epoch 201/1000
58/58 [==============================] - 34s 518ms/step - loss: 0.4796
- accuracy: 0.6915 - val_loss: 0.0794 - val_accuracy: 0.9784
Epoch 202/1000
58/58 [==============================] - 33s 495ms/step - loss: 0.3692
- accuracy: 0.7644 - val_loss: 0.0782 - val_accuracy: 0.9810
Epoch 203/1000
58/58 [==============================] - 34s 508ms/step - loss: 0.4074
- accuracy: 0.7356 - val_loss: 0.0691 - val_accuracy: 0.9810
Epoch 204/1000
58/58 [==============================] - 33s 504ms/step - loss: 0.4397
- accuracy: 0.7035 - val_loss: 0.0855 - val_accuracy: 0.9784
Epoch 205/1000
58/58 [==============================] - 33s 504ms/step - loss: 0.3904
- accuracy: 0.7503 - val_loss: 0.1620 - val_accuracy: 0.9480
Epoch 206/1000
58/58 [==============================] - 33s 504ms/step - loss: 0.4978
- accuracy: 0.6779 - val_loss: 0.1221 - val_accuracy: 0.9619
Epoch 207/1000
58/58 [==============================] - 34s 524ms/step - loss: 0.4129
- accuracy: 0.7247 - val_loss: 0.0785 - val_accuracy: 0.9721
Epoch 208/1000
58/58 [==============================] - 32s 487ms/step - loss: 0.3961
- accuracy: 0.7514 - val_loss: 0.0861 - val_accuracy: 0.9708
Epoch 209/1000
58/58 [==============================] - 34s 515ms/step - loss: 0.4443
- accuracy: 0.7144 - val_loss: 0.0802 - val_accuracy: 0.9721
```

```
Epoch 210/1000
58/58 [==============================] - 33s 499ms/step - loss: 0.4174
- accuracy: 0.7383 - val_loss: 0.0877 - val_accuracy: 0.9708
Epoch 211/1000
58/58 [==============================] - 33s 494ms/step - loss: 0.4114
- accuracy: 0.7231 - val_loss: 0.1088 - val_accuracy: 0.9645
Epoch 212/1000
58/58 [==============================] - 33s 499ms/step - loss: 0.4145
- accuracy: 0.7350 - val_loss: 0.1099 - val_accuracy: 0.9657
Epoch 213/1000
58/58 [==============================] - 33s 490ms/step - loss: 0.4025
- accuracy: 0.7470 - val_loss: 0.0861 - val_accuracy: 0.9670
Epoch 214/1000
58/58 [==============================] - 34s 505ms/step - loss: 0.4837
- accuracy: 0.7122 - val_loss: 0.1860 - val_accuracy: 0.9442
Epoch 215/1000
58/58 [==============================] - 33s 494ms/step - loss: 0.3764
- accuracy: 0.7688 - val_loss: 0.0706 - val_accuracy: 0.9784
Epoch 216/1000
58/58 [==============================] - 32s 481ms/step - loss: 0.3063
- accuracy: 0.8139 - val_loss: 0.0720 - val_accuracy: 0.9759
Epoch 217/1000
58/58 [==============================] - 34s 507ms/step - loss: 0.4189
- accuracy: 0.7301 - val_loss: 0.0913 - val_accuracy: 0.9683
Epoch 218/1000
58/58 [==============================] - 33s 500ms/step - loss: 0.3585
- accuracy: 0.7753 - val_loss: 0.0990 - val_accuracy: 0.9645
Epoch 219/1000
58/58 [==============================] - 35s 526ms/step - loss: 0.5111
- accuracy: 0.6997 - val_loss: 0.1865 - val_accuracy: 0.9315
Epoch 220/1000
58/58 [==============================] - 33s 498ms/step - loss: 0.4566
- accuracy: 0.7029 - val_loss: 0.1355 - val_accuracy: 0.9289
Epoch 221/1000
58/58 [==============================] - 34s 524ms/step - loss: 0.4555
- accuracy: 0.6964 - val_loss: 0.0922 - val_accuracy: 0.9708
Epoch 222/1000
58/58 [==============================] - 34s 507ms/step - loss: 0.4458
- accuracy: 0.6980 - val_loss: 0.0988 - val_accuracy: 0.9721
Epoch 223/1000
58/58 [==============================] - 33s 504ms/step - loss: 0.3759
- accuracy: 0.7557 - val_loss: 0.0895 - val_accuracy: 0.9708
Epoch 00223: early stopping
```

Plotting

In [ ]:
```
loss = history.history['loss']
val_loss = history.history['val_loss']
plt.figure()
plt.plot(loss, 'ro-', label='trn loss')
plt.plot(val_loss, 'bo-' , label='val loss')
plt.ylabel('Cross Entropy')
plt.xlabel('Epoch')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



- the learning stopped at 223 epoch.
- The result
  - trn_loss: 0.3759
  - trn_accuracy: 0.7557
  - val_loss: 0.0895
  - val_accuracy: 0.9708