Физтех-Школа Прикладной математики и информатики (ФПМИ) МФТИ

Ник на каггле Андрей Исупов. Скор 0.99256

Путешествие по Спрингфилду.

Сегодня вам предстоить помочь телекомпании FOX в обработке их контента. Как вы знаете сериал Симсоны идет на телеэкранах более 25 лет и за это время скопилось очень много видео материала. Персоонажи менялись вместе с изменяющимися графическими технологиями и Гомер 2018 не очень похож на Гомера 1989. Нашей задачей будет научиться классифицировать персонажей проживающих в Спрингфилде. Думаю, что нет смысла представлять каждого из них в отдельности.

Установка зависимостей

In [51]:

```
!pip install -U torch torchvision
```

```
Requirement already up-to-date: torch in /usr/local/lib/python3.7/dist
-packages (1.8.1+cu101)
Requirement already up-to-date: torchvision in /usr/local/lib/python3.
7/dist-packages (0.9.1+cu101)
Requirement already satisfied, skipping upgrade: numpy in /usr/local/l
ib/python3.7/dist-packages (from torch) (1.19.5)
Requirement already satisfied, skipping upgrade: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from torch) (3.7.4.3)
Requirement already satisfied, skipping upgrade: pillow>=4.1.1 in /us
r/local/lib/python3.7/dist-packages (from torchvision) (7.1.2)
```

In [52]:

```
# we will verify that GPU is enabled for this notebook
# following should print: CUDA is available!  Training on GPU ...
#
# if it prints otherwise, then you need to enable GPU:
# from Menu > Runtime > Change Runtime Type > Hardware Accelerator > GPU

import torch
import numpy as np

train_on_gpu = torch.cuda.is_available()

if not train_on_gpu:
    print('CUDA is not available.  Training on CPU ...')
else:
    print('CUDA is available!  Training on GPU ...')
```

```
CUDA is available!  Training on GPU ...
```

Type *Markdown* and LaTeX: α^2

In [4]:

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

```
Mounted at /content/gdrive/
```

In [5]:

```
!unzip -q /content/gdrive/MyDrive/train.zip -d train
!unzip -q /content/gdrive/MyDrive/test.zip -d test
```

In [ ]:

```
!ls train
```

```
ls: cannot access 'train': No such file or directory
```

In [6]:

```
!nvidia-smi
import torch
torch.cuda.is_available()
```

```
Sun Apr 25 22:29:08 2021
+-----------------------------------------------------------------------------
--------+
| NVIDIA-SMI 465.19.01    Driver Version: 460.32.03    CUDA Version: 1
1.2     |
|-------------------------------+----------------------+----------------------
--------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Unco
rr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Com
pute M. |
|                               |                      |
MIG M. |
|===============================+======================+======================
========|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |
0 |
| N/A   53C    P8    10W /  70W |      3MiB / 15109MiB |      0%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+----------------------
--------+

+-----------------------------------------------------------------------------
--------+
| Processes:
|
|  GPU   GI   CI        PID   Type   Process name                  GPU
Memory |
|        ID   ID                                                   Usa
ge      |
|=============================================================================
========|
|  No running processes found
|
+-----------------------------------------------------------------------------
--------+
```

Out[6]:

```
True
```

В нашем тесте будет 990 картнок, для которых вам будет необходимо предсказать класс.

In [7]:

```
import pickle
import numpy as np
from skimage import io

from PIL import Image
from pathlib import Path
from tqdm import tqdm, tqdm_notebook

import torch.optim as optim
from torch.optim import lr_scheduler
from torchvision import transforms, models
from multiprocessing.pool import ThreadPool
from sklearn.preprocessing import LabelEncoder
from torch.utils.data import Dataset, DataLoader
import torch.nn as nn

from matplotlib import colors, pyplot as plt
%matplotlib inline

# в sklearn не все гладко, чтобы в colab удобно выводить картинки
# мы будем игнорировать warnings
import warnings
warnings.filterwarnings(action='ignore', category=DeprecationWarning)
```

In [8]:

```
# разные режимы датасета
DATA_MODES = ['train', 'val', 'test']
# все изображения будут масштабированы к размеру 224x224 px
RESCALE_SIZE = 224
# работаем на видеокарте
DEVICE = torch.device("cuda")  # ЗАМЕНИТЬ НА CUDA!1!!!!1!!
```

https://jhui.github.io/2018/02/09/PyTorch-Data-loading-preprocess_torchvision/ (https://jhui.github.io/2018/02/09/PyTorch-Data-loading-preprocess_torchvision/)

Ниже мы исспользуем враппер над датасетом для удобной работы. Вам стоит понимать, что происходит с LabelEncoder и с torch.Transformation.

ToTensor конвертирует PIL Image с параметрами в диапазоне [0, 255] (как все пиксели) в FloatTensor размера (C x H x W) [0,1] , затем производится масштабирование: $input = \frac{input - \mu}{\text{standard deviation}}$, константы - средние и дисперсии по каналам на основе ImageNet

Стоит также отметить, что мы переопределяем метод **getitem** для удобства работы с данной структурой данных. Также используется LabelEncoder для преобразования строковых меток классов в id и обратно. В описании датасета указано, что картинки разного размера, так как брались напрямую с видео, поэтому следуем привести их к одному размер (это делает метод _prepare_sample)

In [53]:

```
class SimpsonsDataset(Dataset):
    """
    Датасет с картинками, который паралельно подгружает их из папок
    производит скалирование и превращение в торчевые тензоры
    """
    def __init__(self, files, mode):
        super().__init__()
        # список файлов для загрузки
        self.files = sorted(files)
        # режим работы
        self.mode = mode

        if self.mode not in DATA_MODES:
            print(f"{self.mode} is not correct; correct modes: {DATA_MODES}")
            raise NameError

        self.len_ = len(self.files)

        self.label_encoder = LabelEncoder()

        if self.mode != 'test':
            self.labels = [path.parent.name for path in self.files]
            self.label_encoder.fit(self.labels)

            with open('label_encoder.pkl', 'wb') as le_dump_file:
                  pickle.dump(self.label_encoder, le_dump_file)

    def __len__(self):
        return self.len_

    def load_sample(self, file):
        image = Image.open(file)
        image.load()
        return image

    def __getitem__(self, index):
        # для преобразования изображений в тензоры PyTorch и нормализации входа
        transform = transforms.Compose([
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        ])
        x = self.load_sample(self.files[index])
        x = self._prepare_sample(x)
        x = np.array(x / 255, dtype='float32')
        x = transform(x)
        if self.mode == 'test':
            return x
        else:
            label = self.labels[index]
            label_id = self.label_encoder.transform([label])
            y = label_id.item()
            return x, y

    def _prepare_sample(self, image):
        image = image.resize((RESCALE_SIZE, RESCALE_SIZE))
        return np.array(image)
```

In [54]:

```
def imshow(inp, title=None, plt_ax=plt, default=False):
    """Imshow для тензоров"""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt_ax.imshow(inp)
    if title is not None:
        plt_ax.set_title(title)
    plt_ax.grid(False)
```

In [55]:

```
TRAIN_DIR = Path('./train/train/simpsons_dataset/')
TEST_DIR = Path('./test/testset/')

train_val_files = sorted(list(TRAIN_DIR.rglob('*.jpg')))
test_files = sorted(list(TEST_DIR.rglob('*.jpg')))
```

In [55]:

```

```

In [56]:

```
from sklearn.model_selection import train_test_split

train_val_labels = [path.parent.name for path in train_val_files]
train_files, val_files = train_test_split(train_val_files, test_size=0.25, \
                                          stratify=train_val_labels)
```

In [57]:

```
val_dataset = SimpsonsDataset(val_files, mode='val')
```

In [58]:

```
len(np.unique(train_val_labels))
len(train_val_labels)
```

Out[58]:

```
20933
```

Давайте посмотрим на наших героев внутри датасета.

In [60]:

```
fig, ax = plt.subplots(nrows=3, ncols=3,figsize=(8, 8), \
                       sharey=True, sharex=True)
for fig_x in ax.flatten():
    random_characters = int(np.random.uniform(0,1000))
    im_val, label = val_dataset[random_characters]
    img_label = " ".join(map(lambda x: x.capitalize(),\
                val_dataset.label_encoder.inverse_transform([label])[0].split('_')))
    imshow(im_val.data.cpu(), \
          title=img_label,plt_ax=fig_x)
```

Можете добавить ваши любимые сцены и классифицировать их. (веселые результаты можно кидать в чат)

In [61]:

```
def fit_epoch(model, train_loader, criterion, optimizer):
    running_loss = 0.0
    running_corrects = 0
    processed_data = 0

    for inputs, labels in train_loader:
        inputs = inputs.to(DEVICE)
        labels = labels.to(DEVICE)
        optimizer.zero_grad()

        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        preds = torch.argmax(outputs, 1)
        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)
        processed_data += inputs.size(0)

    train_loss = running_loss / processed_data
    train_acc = running_corrects.cpu().numpy() / processed_data
    return train_loss, train_acc
```

In [62]:

```
def eval_epoch(model, val_loader, criterion):
    model.eval()
    running_loss = 0.0
    running_corrects = 0
    processed_size = 0

    for inputs, labels in val_loader:
        inputs = inputs.to(DEVICE)
        labels = labels.to(DEVICE)

        with torch.set_grad_enabled(False):
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            preds = torch.argmax(outputs, 1)

        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)
        processed_size += inputs.size(0)
    val_loss = running_loss / processed_size
    val_acc = running_corrects.double() / processed_size
    return val_loss, val_acc
```

In [63]:

```
def train(train_files, val_files, model, opt, criterion, epochs, batch_size):
    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

    best_model_wts = model.state_dict()
    best_acc = 0.0
    history = []
    log_template = "\nEpoch {ep:03d} train_loss: {t_loss:0.4f} \
    val_loss {v_loss:0.4f} train_acc {t_acc:0.4f} val_acc {v_acc:0.4f}"

    with tqdm(desc="epoch", total=epochs) as pbar_outer:

        for epoch in range(epochs):
            train_loss, train_acc = fit_epoch(model, train_loader, criterion, opt)
            print("loss", train_loss)

            val_loss, val_acc = eval_epoch(model, val_loader, criterion)
            history.append((train_loss, train_acc, val_loss, val_acc))

            pbar_outer.update(1)
            tqdm.write(log_template.format(ep=epoch+1, t_loss=train_loss,\
                                           v_loss=val_loss, t_acc=train_acc, v_acc=

            # если достиглось лучшее качество, то запомним веса модели
            if val_acc > best_acc:
                best_acc = val_acc
                best_model_wts = model.state_dict()

    # загрузим лучшие веса модели
    model.load_state_dict(best_model_wts)

    return model, history
```

In [64]:

```
def predict(model, test_loader):
    with torch.no_grad():
        logits = []

        for inputs in test_loader:
            inputs = inputs.to(DEVICE)
            model.eval()
            outputs = model(inputs).cpu()
            logits.append(outputs)

    probs = nn.functional.softmax(torch.cat(logits), dim=-1).numpy()
    return probs
```

In [65]:

```
n_classes = len(np.unique(train_val_labels))

# Качаем модель
model = models.resnet50(pretrained=True)
print(model)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=
(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track
_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=
1, ceil_mode=False)
  (layer1): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bia
s=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, t
rack_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), pad
ding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, t
rack_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bi
```

In [66]:

```
len(np.unique(train_val_labels))
```

Out[66]:

```
42
```

In [67]:

```
for i, (name,param) in enumerate(model.named_parameters()):
    print(i, name)
# len(list(model.parameters()))
```

```
0 conv1.weight
1 bn1.weight
2 bn1.bias
3 layer1.0.conv1.weight
4 layer1.0.bn1.weight
5 layer1.0.bn1.bias
6 layer1.0.conv2.weight
7 layer1.0.bn2.weight
8 layer1.0.bn2.bias
9 layer1.0.conv3.weight
10 layer1.0.bn3.weight
11 layer1.0.bn3.bias
12 layer1.0.downsample.0.weight
13 layer1.0.downsample.1.weight
14 layer1.0.downsample.1.bias
15 layer1.1.conv1.weight
16 layer1.1.bn1.weight
17 layer1.1.bn1.bias
18 layer1.1.conv2.weight
```

In [68]:

```
# Замораживаем градиенты
layers_to_unfreeze = 89
for param in list(model.parameters())[:-layers_to_unfreeze]:
    param.requires_grad = False

# Меняем классификатор
num_features = 2048
n_classes = len(np.unique(train_val_labels))
model.classifier = nn.Sequential(
    nn.Dropout(p=0.4),
    nn.Linear(num_features, 1024),
    nn.ReLU(),
    nn.Dropout(p=0.4),
    nn.Linear(1024, 1024),
    nn.ReLU(),
    nn.Linear(1024, n_classes)
)

# Переводим на gpu
if train_on_gpu:
    model = model.cuda()

# Loss function
loss_fn = nn.CrossEntropyLoss()

# Оптимизатор
params_to_update = []
for name,param in model.named_parameters():
    if param.requires_grad == True:
        params_to_update.append(param)
        print("\t",name)
optimizer = optim.AdamW(params_to_update, lr=1e-4)

# Динамический learning rate
exp_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.5)
```

```
         layer3.0.conv1.weight
         layer3.0.bn1.weight
         layer3.0.bn1.bias
         layer3.0.conv2.weight
         layer3.0.bn2.weight
         layer3.0.bn2.bias
         layer3.0.conv3.weight
         layer3.0.bn3.weight
         layer3.0.bn3.bias
         layer3.0.downsample.0.weight
         layer3.0.downsample.1.weight
         layer3.0.downsample.1.bias
         layer3.1.conv1.weight
         layer3.1.bn1.weight
         layer3.1.bn1.bias
         layer3.1.conv2.weight
         layer3.1.bn2.weight
         layer3.1.bn2.bias
         layer3.1.conv3.weight
         layer3.1.bn3.weight
         layer3.1.bn3.bias
         layer3.2.conv1.weight
         layer3.2.bn1.weight
```

```
layer3.2.bn1.bias
layer3.2.conv2.weight
layer3.2.bn2.weight
layer3.2.bn2.bias
layer3.2.conv3.weight
layer3.2.bn3.weight
layer3.2.bn3.bias
layer3.3.conv1.weight
layer3.3.bn1.weight
layer3.3.bn1.bias
layer3.3.conv2.weight
layer3.3.bn2.weight
layer3.3.bn2.bias
layer3.3.conv3.weight
layer3.3.bn3.weight
layer3.3.bn3.bias
layer3.4.conv1.weight
layer3.4.bn1.weight
layer3.4.bn1.bias
layer3.4.conv2.weight
layer3.4.bn2.weight
layer3.4.bn2.bias
layer3.4.conv3.weight
layer3.4.bn3.weight
layer3.4.bn3.bias
layer3.5.conv1.weight
layer3.5.bn1.weight
layer3.5.bn1.bias
layer3.5.conv2.weight
layer3.5.bn2.weight
layer3.5.bn2.bias
layer3.5.conv3.weight
layer3.5.bn3.weight
layer3.5.bn3.bias
layer4.0.conv1.weight
layer4.0.bn1.weight
layer4.0.bn1.bias
layer4.0.conv2.weight
layer4.0.bn2.weight
layer4.0.bn2.bias
layer4.0.conv3.weight
layer4.0.bn3.weight
layer4.0.bn3.bias
layer4.0.downsample.0.weight
layer4.0.downsample.1.weight
layer4.0.downsample.1.bias
layer4.1.conv1.weight
layer4.1.bn1.weight
layer4.1.bn1.bias
layer4.1.conv2.weight
layer4.1.bn2.weight
layer4.1.bn2.bias
layer4.1.conv3.weight
layer4.1.bn3.weight
layer4.1.bn3.bias
layer4.2.conv1.weight
layer4.2.bn1.weight
layer4.2.bn1.bias
layer4.2.conv2.weight
layer4.2.bn2.weight
layer4.2.bn2.bias
```

```
layer4.2.conv3.weight
layer4.2.bn3.weight
layer4.2.bn3.bias
fc.weight
fc.bias
classifier.1.weight
classifier.1.bias
classifier.4.weight
classifier.4.bias
classifier.6.weight
classifier.6.bias
```

Запустим обучение сети.

In [69]:

```
if val_dataset is None:
    val_dataset = SimpsonsDataset(val_files, mode='val')

train_dataset = SimpsonsDataset(train_files, mode='train')
```

In [70]:

```
model, history = train(train_dataset, val_dataset, model, optimizer, loss_fn, epoch
```

```
epoch:   0%|          | 0/25 [00:00<?, ?it/s]

loss 0.8504349717800234

epoch:   4%|▍         | 1/25 [03:52<1:32:53, 232.25s/it]


Epoch 001 train_loss: 0.8504     val_loss 0.1921 train_acc 0.8482 va
l_acc 0.9530
loss 0.21007397585969717

epoch:   8%|▊         | 2/25 [07:38<1:28:20, 230.47s/it]


Epoch 002 train_loss: 0.2101     val_loss 0.2416 train_acc 0.9459 va
l_acc 0.9412
loss 0.08366544071436156

epoch:  12%|█▏        | 3/25 [11:23<1:23:55, 228.91s/it]
```

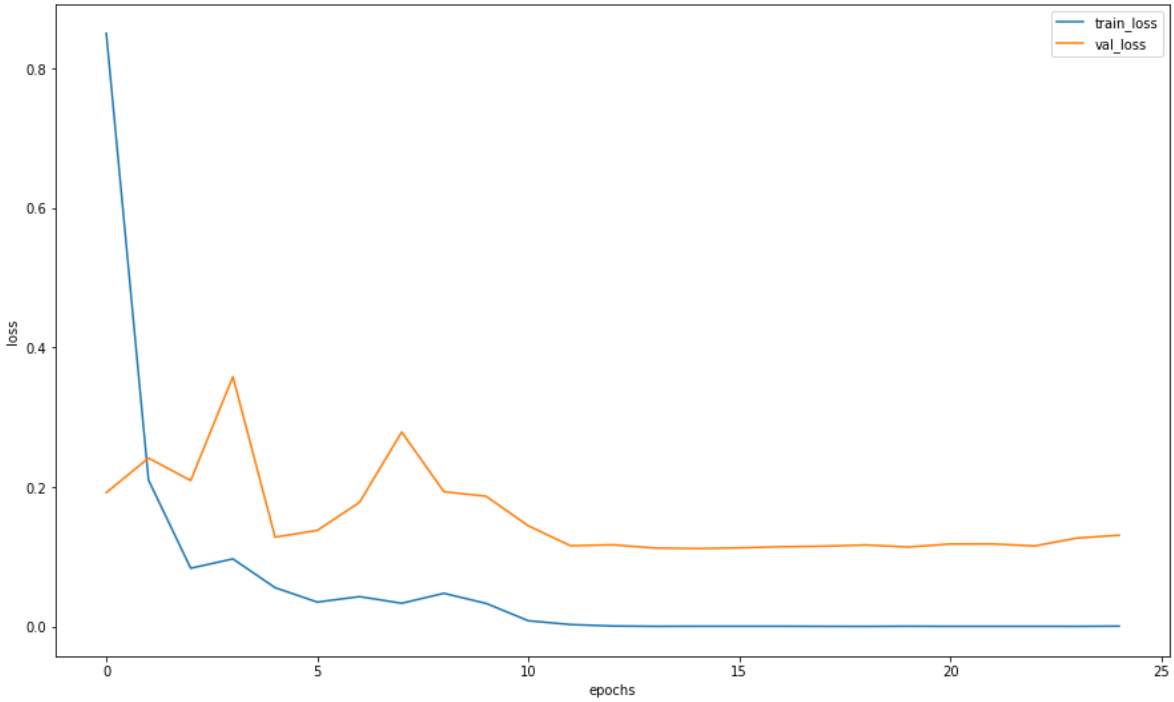
Построим кривые обучения

In [71]:

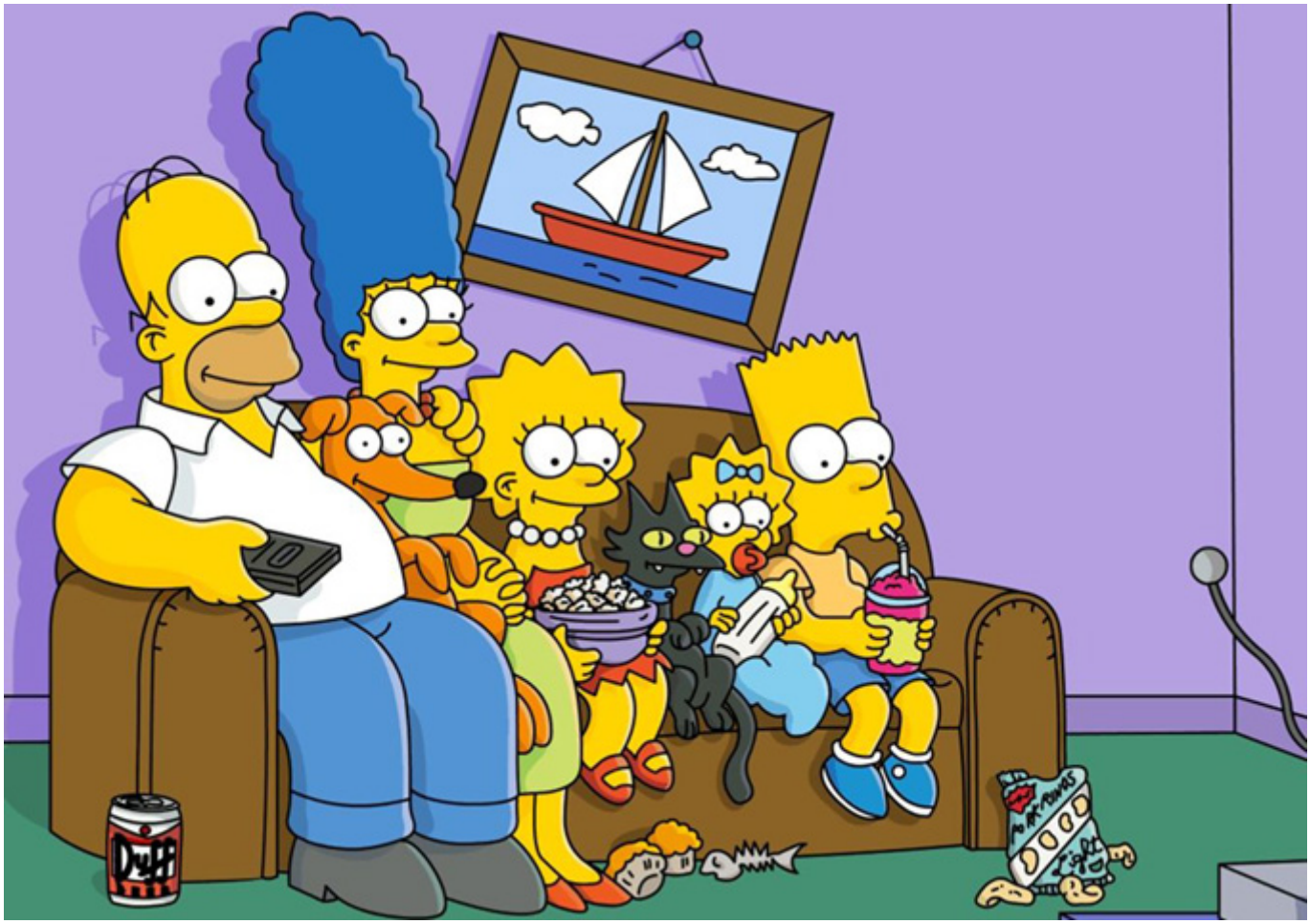
```
loss, acc, val_loss, val_acc = zip(*history)
```

In [72]:

```
plt.figure(figsize=(15, 9))
plt.plot(loss, label="train_loss")
plt.plot(val_loss, label="val_loss")
plt.legend(loc='best')
plt.xlabel("epochs")
plt.ylabel("loss")
plt.show()
```



Ну и что теперь со всем этим делать?



Хорошо бы понять, как сделать сабмит. У нас есть сеть и методы eval у нее, которые позволяют перевести сеть в режим предсказания. Стоит понимать, что у нашей модели на последнем слое стоит softmax, которые позволяет получить вектор вероятностей того, что объект относится к тому или иному классу. Давайте воспользуемся этим.

In [73]:

```
def predict_one_sample(model, inputs, device=DEVICE):
    """Предсказание, для одной картинки"""
    with torch.no_grad():
        inputs = inputs.to(device)
        model.eval()
        logit = model(inputs).cpu()
        probs = torch.nn.functional.softmax(logit, dim=-1).numpy()
    return probs
```

In [74]:

```
random_characters = int(np.random.uniform(0,1000))
ex_img, true_label = val_dataset[random_characters]
probs_im = predict_one_sample(model, ex_img.unsqueeze(0))
```

In [75]:

```
idxs = list(map(int, np.random.uniform(0,250, 20)))
imgs = [val_dataset[id][0].unsqueeze(0) for id in idxs]

probs_ims = predict(model, imgs)
```

In [76]:

```
label_encoder = pickle.load(open("label_encoder.pkl", 'rb'))
```

In [77]:

```
y_pred = np.argmax(probs_ims,-1)

actual_labels = [val_dataset[id][1] for id in idxs]

preds_class = [label_encoder.classes_[i] for i in y_pred]
```

Обратите внимание, что метрика, которую необходимо оптимизировать в конкурсе --- f1-score. Вычислим целевую метрику на валидационной выборке.

In [78]:

```
from sklearn.metrics import f1_score

f1_score(actual_labels, y_pred, average=None)
```

Out[78]:

```
array([0.96969697, 1.        , 0.        ])
```

Сделаем классную визуализацию, чтобы посмотреть насколько сеть уверена в своих ответах. Можете исспользовать это, чтобы отлаживать правильность вывода.

In [79]:

```
import matplotlib.patches as patches
from matplotlib.font_manager import FontProperties

fig, ax = plt.subplots(nrows=3, ncols=3,figsize=(12, 12), \
                       sharey=True, sharex=True)
for fig_x in ax.flatten():
    random_characters = int(np.random.uniform(0,250))
    im_val, label = val_dataset[random_characters]
    img_label = " ".join(map(lambda x: x.capitalize(),\
                val_dataset.label_encoder.inverse_transform([label])[0].split('_'))



    imshow(im_val.data.cpu(), \
          title=img_label,plt_ax=fig_x)

    actual_text = "Actual : {}".format(img_label)

    fig_x.add_patch(patches.Rectangle((0, 53),86,35,color='white'))
    font0 = FontProperties()
    font = font0.copy()
    font.set_family("fantasy")
    prob_pred = predict_one_sample(model, im_val.unsqueeze(0))
    predicted_proba = np.max(prob_pred)*100
    y_pred = np.argmax(prob_pred)

    predicted_label = label_encoder.classes_[y_pred]
    predicted_label = predicted_label[:len(predicted_label)//2] + '\n' + predicted_
    predicted_text = "{} : {:.0f}%".format(predicted_label,predicted_proba)

    fig_x.text(1, 59, predicted_text , horizontalalignment='left', fontproperties=f
                    verticalalignment='top',fontsize=8, color='black',fontweight='b
```

Попробуйте найти те классы, которые сеть не смогла расспознать. Изучите данную проблему, это понадобится в дальнейшем.

Submit на Kaggle

In [80]:

```
len(test_files)
```

Out[80]:

```
991
```

In [81]:

```
test_dataset = SimpsonsDataset(test_files, mode="test")
test_loader = DataLoader(test_dataset, shuffle=False, batch_size=64)
probs = predict(model, test_loader)


preds = label_encoder.inverse_transform(np.argmax(probs, axis=1))
test_filenames = [path.name for path in test_dataset.files]
```

In [47]:

```
! ls
```

```
gdrive  label_encoder.pkl  sample_data  test  train
```

In [82]:

```
probs = predict(model, test_loader)

preds = label_encoder.inverse_transform(np.argmax(probs, axis=1))
test_filenames = [path.name for path in test_dataset.files]
```

In [83]:

```
import pandas as pd

my_submit = pd.DataFrame({'Id': test_filenames, 'Expected': preds})
my_submit['ind'] = my_submit['Id'].apply(lambda x: x[3:-4])
my_submit['ind'] = my_submit['ind'].astype('int')
my_submit.sort_values(by='ind', inplace=True)
my_submit.reset_index(inplace=True, drop=True)
my_submit.drop(columns=['ind'], inplace=True, axis=1)

my_submit.head(25)
```

Out[83]:

	Id	Expected
0	img0.jpg	nelson_muntz
1	img1.jpg	bart_simpson
2	img2.jpg	mayor_quimby
3	img3.jpg	nelson_muntz
4	img4.jpg	lisa_simpson
5	img5.jpg	principal_skinner
6	img6.jpg	krusty_the_clown
7	img7.jpg	apu_nahasapeemapetilon
8	img8.jpg	principal_skinner
9	img9.jpg	comic_book_guy
10	img10.jpg	ned_flanders
11	img11.jpg	ned_flanders
12	img12.jpg	homer_simpson
13	img13.jpg	abraham_grampa_simpson
14	img14.jpg	nelson_muntz
15	img15.jpg	charles_montgomery_burns
16	img16.jpg	mayor_quimby
17	img17.jpg	comic_book_guy
18	img18.jpg	homer_simpson
19	img19.jpg	milhouse_van_houten
20	img20.jpg	marge_simpson
21	img21.jpg	principal_skinner
22	img22.jpg	moe_szyslak
23	img23.jpg	chief_wiggum
24	img24.jpg	chief_wiggum

In [84]:

```
# my_submit.to_csv('./simple_cnn_baseline.csv', index=False)
my_submit.to_csv('gdrive/My Drive/ResNet2_submission.csv', index=False)
```

In [85]:

```
torch.save(model.state_dict(), 'gdrive/My Drive/ResNet2.pth')
```

Приключение?

А теперь самое интересное, мы сделали простенькую сверточную сеть и смогли отправить сабмит, но получившийся скор нас явно не устраивает. Надо с этим что-то сделать.

Несколько срочныйх улучшейни для нашей сети, которые наверняка пришли Вам в голову:

- Учим дольше и изменяем гиперпараметры сети
- learning rate, batch size, нормализация картинки и вот это всё
- Кто же так строит нейронные сети? А где пулинги и батч нормы? Надо добавлять
- Ну разве Адам наше все? adamW (https://www.fast.ai/2018/07/02/adam-weight-decay/) для практика, статейка для любителей (https://openreview.net/pdf?id=ryQu7f-RZ) (очень хороший анализ), наши (https://github.com/MichaelKonobeev/adashift/) эксперименты для заинтересованных.
- Ну разве это deep learning? Вот ResNet и Inception, которые можно зафайнтьюнить под наши данные, вот это я понимаю (можно и обучить в колабе, а можно и готовые (https://github.com/Cadene/pretrained-models.pytorch) скачать).
- Данных не очень много, можно их аугументировать и доучититься на новом датасете (который уже будет состоять из, как пример аугументации, перевернутых изображений)
- Стоит подумать об ансамблях

Надеюсь, что у Вас получится!

In [ ]:

```

```