Artificial Intelligence and Machine Learning, Homework #2

Marco Testa, s265861

December 17, 2019

Contents

1 Introduction

This report refers the second Homework of the Machine Learning and Artificial Intelligence course at PoliTo. It focuses on the image classification by training the CNN AlexNet on Caltech-101 Dataset.

1.1 DataSet description

The DataSet we are using is Caltech-101 containing pictures of objects belonging to 101 categories. There are about 40 to 800 images per category and most categories have about 50 images. The original size of each image is roughly 300x200 pixels.

2 Data Preprocessing

In the GitHub repository https://github.com/MachineLearning2020/Homework2-Caltech101 two txt files are given. One contains the list of the paths for the training images, the other the list for the test ones. Also the lists include the BACKGROUND_Google category which has to be removed. For this a class very similar to pytorch datasetFolder class has been created to handle this.

2.1 Caltech class

Caltech class is a class created for two reasons:

1. For reading the split files
2. For filtering out the unwanted class It aims to create the dataset objects.

Filtering is at level of : make_dataset() and _find_classes() methods

```
  ...
def make_dataset(dir, class_to_idx, extensions=None, is_valid_file=
                                    None, split=None):
    #set_trace()
    dir = os.path.expanduser(dir)
    home = dir.split('/') # home = Homework2-Caltech101
    images = []
    #set_trace()

    if not ((extensions is None) ^ (is_valid_file is None)):
        raise ValueError("Both extensions and is_valid_file cannot
                                        be None or not None at
                                        the same time")
    if extensions is not None:
        def is_valid_file(x):
            return has_file_allowed_extension(x, extensions)

    if split == 'train':
        train_images_paths = [line.rstrip('\n') for line in open(
                                        home[0]+'/train.txt')  if
                                         'BACKGROUND_Google/' not
                                         in line]
        ...
```

```
  ...
def _find_classes(self, dir):
  classes = []
        #set_trace()
        home = dir.split('/')[0]
        train_images_paths = [line.rstrip('\n') for line in open(
                                        home+'/train.txt')  if '
                                        BACKGROUND_Google/' not
                                        in line]
  ...
```

Where home is 'Homework2-Caltech101', and so the 101 classes have been known by the two files .txt.

3 Training

3.1 Validation Set

Training size is 5784, Test size, ValidationSet is retrieved from half of the TrainingSet which is halved. I create indexes for training and validation from the entire TrainingSet object, which is not really splitted at beginning. Then applying Subset() the two datasets are created.

```
val_split = .5
shuffle_dataset = True
random_seed = 20
#Creating data indexes for training and validation splits:

training_big_size = len(train_dataset)
indexes = list(range(training_big_size))
split = int(np.floor(val_split * training_big_size))


if shuffle_dataset:
  np.random.seed(random_seed)
  np.random.shuffle(indexes)
train_idx, val_idx = indexes[split:], indexes[:split]

train_dataset_new = Subset(train_dataset, train_idx)
val_dataset_new = Subset(train_dataset, val_idx)
print('Index for new training and validation set created:\nTraining
                                : {}\nValidation: {}'.format(
                                sorted(train_idx), sorted(val_idx
                                )))
print('New sizes:\ntrain_new: {}\nval_new: {}'.format(len(
                                train_dataset_new), len(
                                val_dataset_new)))
```

3.2 Model Selection

Having our Training sand Validation sets, now we are going to evaluate the model on the ValidationSet trying different sets of hyperparameters. The set which will give us the best accuracy on ValidationSet is chosen for testing later on TestSet. The parameters I want to tune are:

1. Learning Rate [default = 0.001]
2. Number of epochs [default = 30]
3. Gamma [default = 0.1]

3.2.1 First set of hyperparameters

The parameters are in default values.

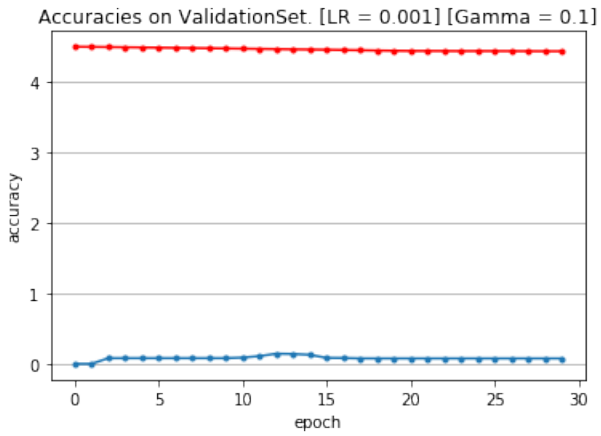

Figure 1: Default parameters

We obtain as the best accuracy value 15.38% for the 13th epoch with related loss of 4.58.

3.3 Second set of hyperparameters

Changed parameters:

1. Learning Rate = from 0.001 to 0.1

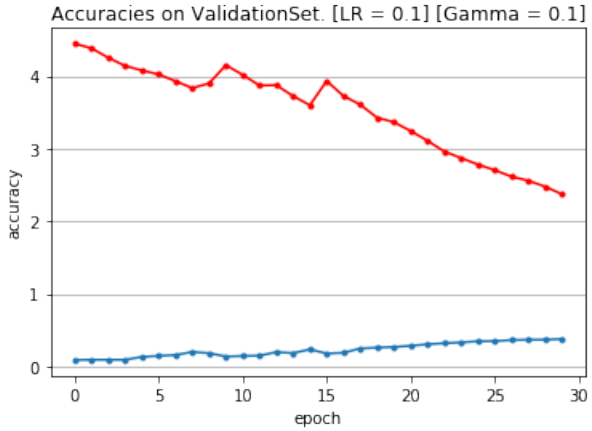

Figure 2: LearningRate: 0.1

The best accuracy is for the 30th epoch, with a value corresponding to 37% with related loss of 2.199 At the 30th epoch the LR reached 0.01, which

suggests that probably we are finding for this value of learning rate with this set of parameters.

3.4 Third set of hyperparameters

1. Learning Rate = from 0.1 to 0.01
2. Gamma = 0.2

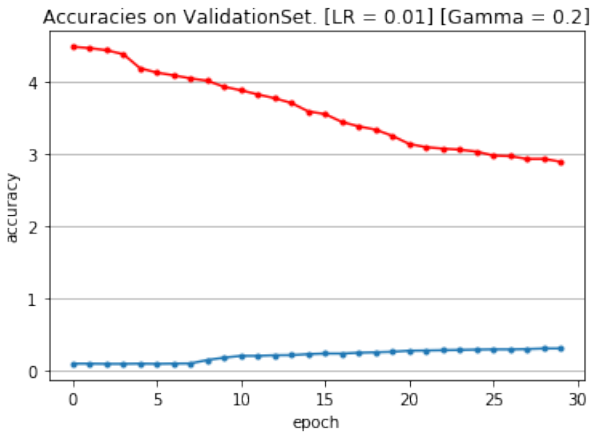

Figure 3: LearningRate: 0.01

We can see that Loss decreases in a smoother way than before, but values of accuracies and loss itself are less performant than before.

3.5 Fourth set of hyperparameters

1. Learning Rate = 0.01
2. Gamma = from 0.2 to 0.8

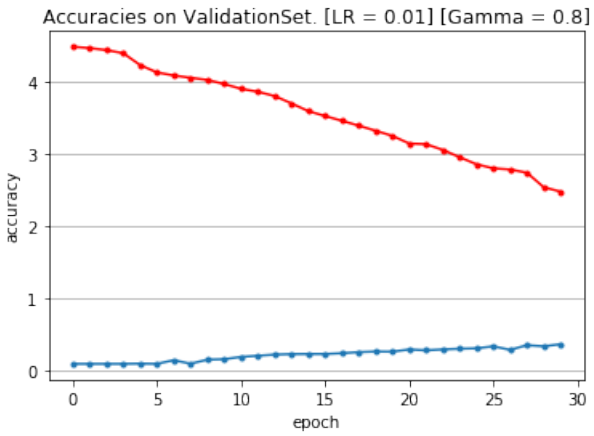

Figure 4: LearningRate: 0.01, Gamma: 0.8

Increasing gamma until 0.8 the model reach almost the same performance than the model with an higher learning rate. This could be a good trade-off.

3.6 Comparisons

Table 1: Comparison for model selection

Set	*Validation*	*Test*	*Loss*
Set1	9.58%	10.51%	4.54
Set2	37.62%	38.4%	2.19
Set3	30.11%	32.21%	3.10
Set4	36.03%	37.60%	2.69

From the table which shows the different accuracies on different sets for the four different set of hyperparameters. The model chosen is the one characterized by the Set4 because of the Learning rate not so high (0.1), but higher than the default value (0.001). The accuracy on TestSet of Set2 is slightly better than Set4, but I decide to keep Set4 configuration.

4 Transfer Learning

4.1 Intro

In this section we are going to use a pre-trained version of AlexNet, trained on ImageNet. For this reason we have to change the mean and variance values used in the Normalize() function. Invoque alexnet in this way:

```
net = alexnet(pretrained=True)
```

4.2 First set of hyperparameters

The first set of this experiment is equal to the set in section 3.3.

1. Learning Rate = 0.1

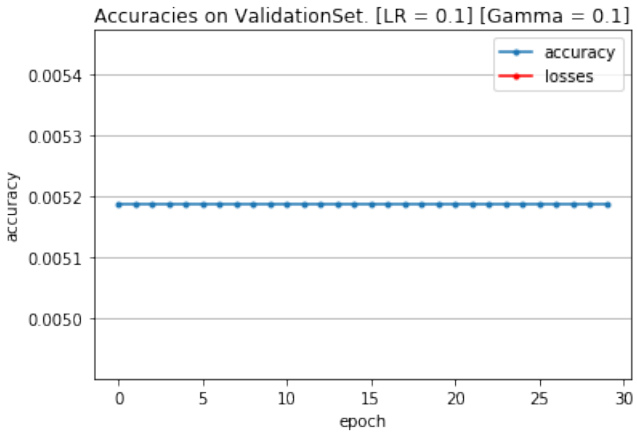

Figure 5: LearningRate: 0.1

The loss has value nan, so it explodes. This lead us to consider this as a bad configuration.

4.3 Second set of hyperparameters

1. Learning Rate = 0.01
2. Gamma = 0.2

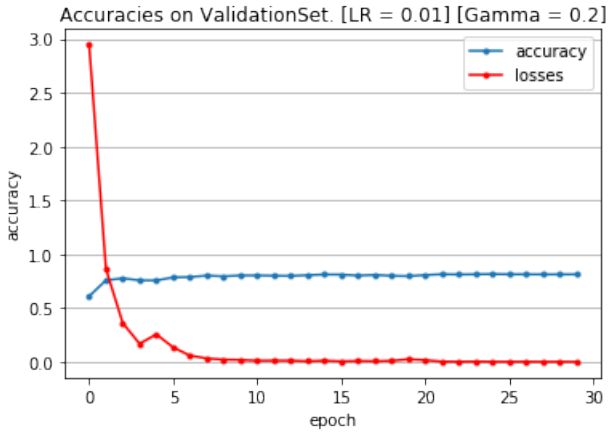

Figure 6: LearningRate: 0.1, Gamma: 0.2

4.4 Third set of hyperparameters

1. Learning Rate = 0.01
2. Gamma = 0.8

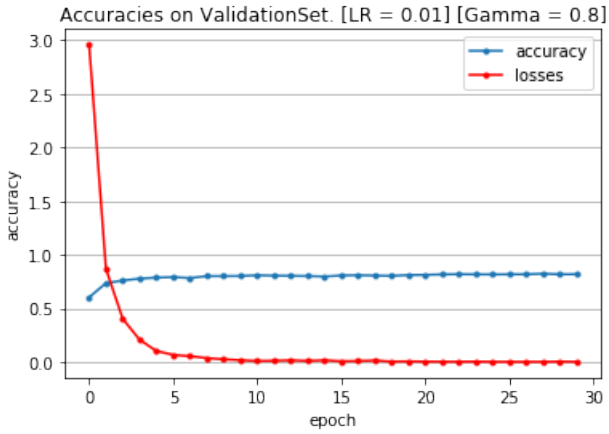

Figure 7: LearningRate: 0.01, Gamma: 0.8

4.5 Comparisons

Definitely the best Set for the model is Set3, we keep this configuration for the next points.

Table 2: Comparison for model selection

Set	*Validation*	*Test*	*Loss*
Set1	0.5%	0.6%	nan
Set2	81.32%	84.20%	0.001
Set3	82.33%	85.41%	0.0006

4.6 Training only Fully Connected Layers

In this subsection we are going to train only the Fully Connected Layers in order to be faster. The optimizer will not receive all parameters of the net, but only net.features.parameters() The results are the following:

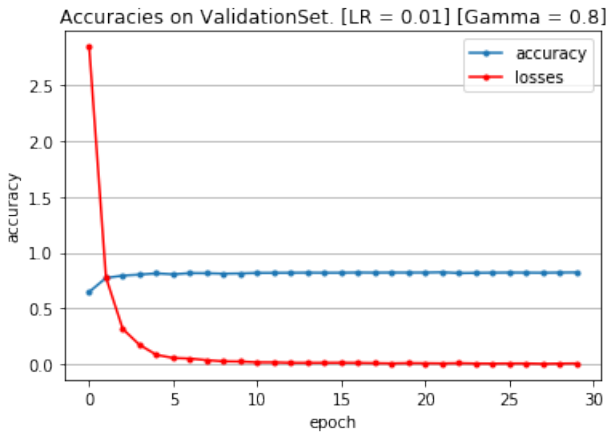

Figure 8: LearningRate: 0.01, Gamma: 0.8

The accuracy on ValidationSet is 82.45% with loss = 0.006. The accuracy on the TestSet instead is 85.13%.

4.7 Training only Convolutional Layers

In this subsection we are going to train only the Convolutional Layers in order to be faster. The optimizer will not receive all parameters of the net, but only net.classifier.parameters() The results are the following:

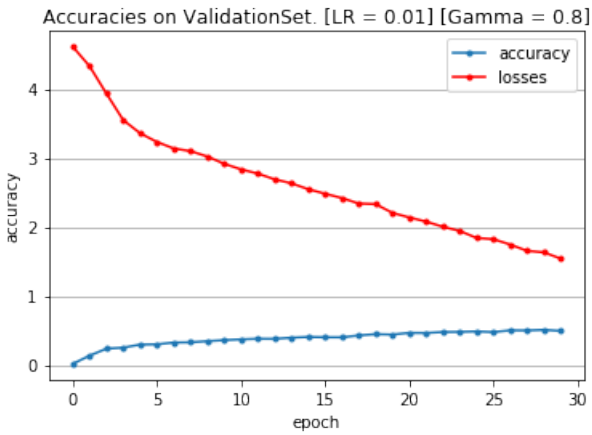

Figure 9: LearningRate: 0.01, Gamma: 0.8

The accuracy on ValidationSet is 50.89% with loss = 1.617. The accuracy on the TestSet instead is 51.91%.

4.7.1 Comparison

Table 3: Comparison

Freeze	*Validation*	*Test*	*Loss*
CL	82.45%	85.13%	0.006
FC	50.89%	51.91%	1.617

As we can see when we train only the Convolutional Layers our performance degrade a lot. Convolutional Layers extract features from the dataset, so training only these layers will affect features learned in the pretraining phase. While Fully Connected Layers don't extract features, but they classify the dataset based on the features learned before. The results show this difference pretty well.

5 Data Augmentation

5.1 Intro

In this section we are going to use a pre-trained version of AlexNet, trained on ImageNet. For this reason we have to change the mean and variance values used in the Normalize() function. Invoque alexnet in this way:

```
net = alexnet(pretrained=True)
```

5.2 First set of transformations

This is the first composition of transformations for data augmentation we are going to use on the pretrained Alexnet.

```
train_aug_transform_1 = transforms.Compose([transforms.Resize(256),
                                 transforms.ColorJitter(
                                brightness=20, contrast=10,
                                saturation=30),  transforms.
                                CenterCrop(224), transforms.
                                ToTensor(), transforms.Normalize
                                ((0.485, 0.456, 0.406), (0.229, 0
                                .224, 0.225))])

eval_aug_transform_1 = transforms.Compose([transforms.Resize(256),
                                transforms.ColorJitter(brightness
                                =20, contrast=10, saturation=30),
                                 transforms.CenterCrop(224),
                                transforms.ToTensor(), transforms
                                .Normalize((0.485, 0.456, 0.406),
                                 (0.229, 0.224, 0.225))])
```

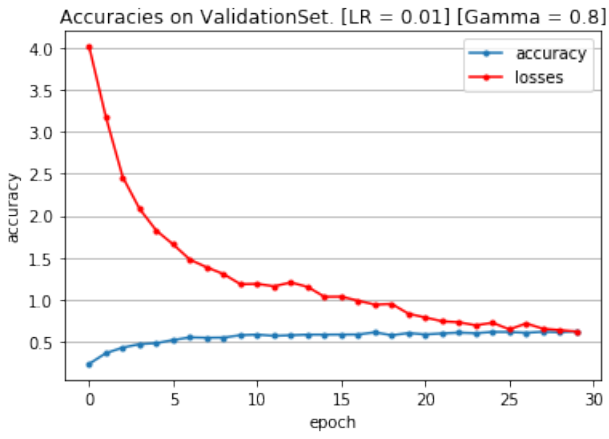

Figure 10: TransformSet1

The Accuracy on the ValidationSet is 62.44%, with loss: 0.57 and the Accuracy on the TestSet is 63.74%

5.3 Second set of transformations

```
train_aug_transform_2 = transforms.Compose([transforms.Resize(256),
                                    transforms.ColorJitter(contrast=
                                    10, hue=0.2),  transforms.
                                    RandomHorizontalFlip(p=0.5),
                                    transforms.CenterCrop(224),
                                    transforms.ToTensor(),
                                    transforms.Normalize((0.485, 0.
                                    456, 0.406), (0.229, 0.224, 0.225
                                    ))])

eval_aug_transform_2 = transforms.Compose([transforms.Resize(256),
                                    transforms.ColorJitter(contrast=
                                    10, hue=0.2),  transforms.
                                    RandomHorizontalFlip(p=0.5),
                                    transforms.CenterCrop(224),
                                    transforms.ToTensor(), transforms
                                    .Normalize((0.485, 0.456, 0.406),
                                     (0.229, 0.224, 0.225))])
```

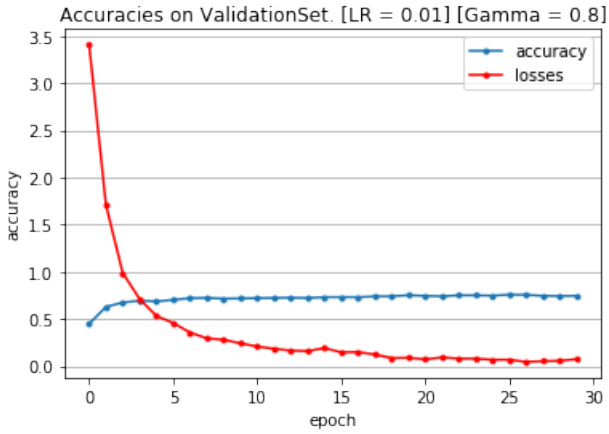

Figure 11: TransformSet2

The Accuracy on the ValidationSet is 75.69%, with loss: 0.03 and the Accuracy on the TestSet is 77.63%

5.4 Third set of transformations

```
train_aug_transform_3 = transforms.Compose([transforms.Resize(256),
                                            transforms.RandomRotation(
                                           degrees=15), transforms.
                                           CenterCrop(224), transforms.
                                           ToTensor(), transforms.Normalize
                                           ((0.485, 0.456, 0.406), (0.229, 0
                                           .224, 0.225))
])

eval_aug_transform_3 = transforms.Compose([transforms.Resize(256),
                                           transforms.RandomRotation(degrees
                                           =15), transforms.CenterCrop(224),
                                            transforms.ToTensor(),
                                           transforms.Normalize((0.485, 0.
                                           456, 0.406), (0.229, 0.224, 0.225
                                           ))
])
```

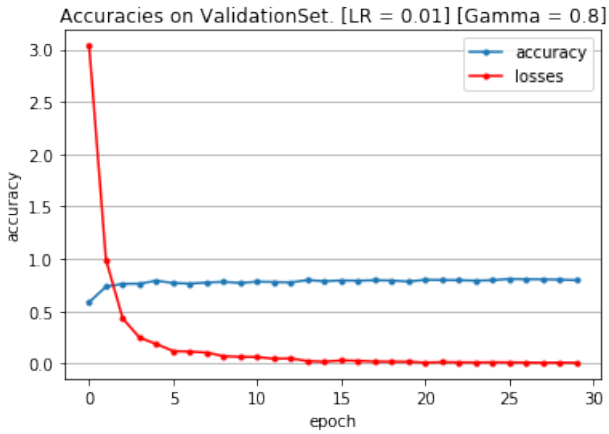

Figure 12: LearningRate: 0.01, Gamma: 0.8

5.5 Comparisons

Table 4: Comparison between Transforms sets

Transform	*Validation*	*Test*	*Loss*
Set1	62.44%	63.74%	0.57
Set2	75.69%	77.63%	0.03
Set3	80.77%	82.54%	0.004

At the end of the three run with these three different sets of Transformations we see how the Set3 is the best for final perfomances, but it is still less than the DefaultSet of transformations.