# LAB 2-1 : Use simple data augmentation to improve the performance of your model

## Without training data augmentation

### Source code of data transformation

data\_transforms = {

'train': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

transforms.ToTensor(),

transforms.Normalize([0.5548, 0.4508, 0.3435], [0.2281, 0.2384, 0.2376])

]),

'val': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

'test': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

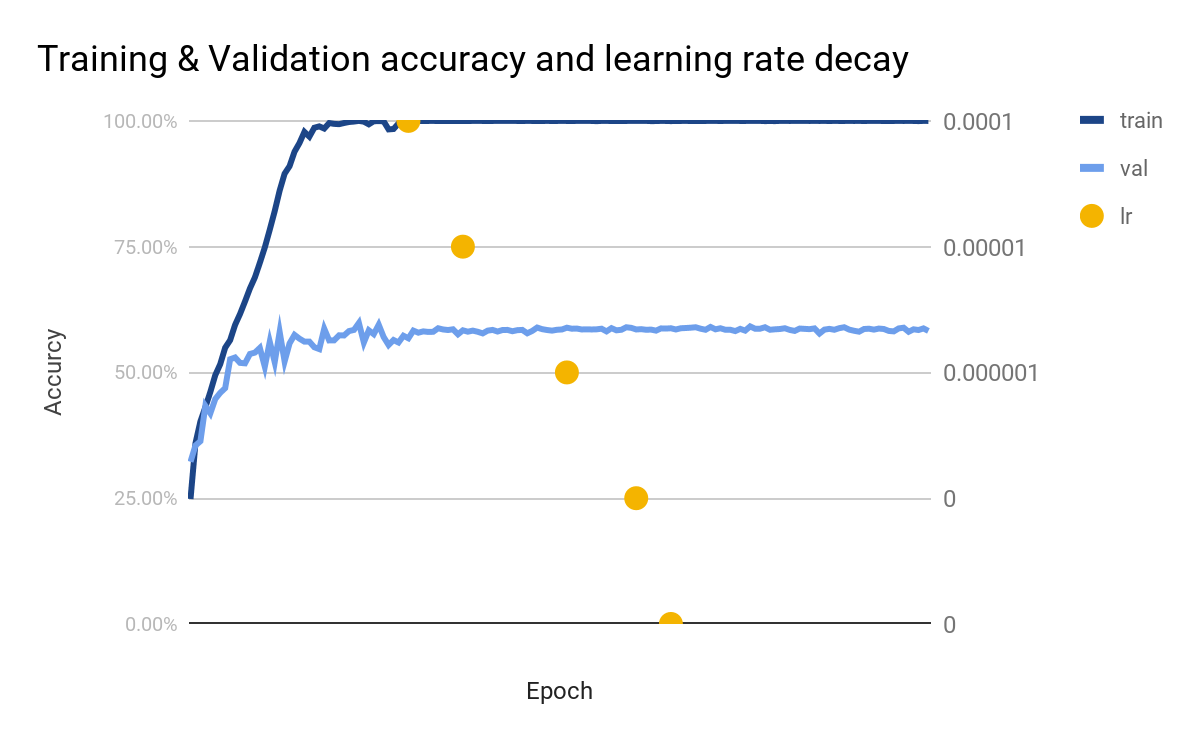
transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

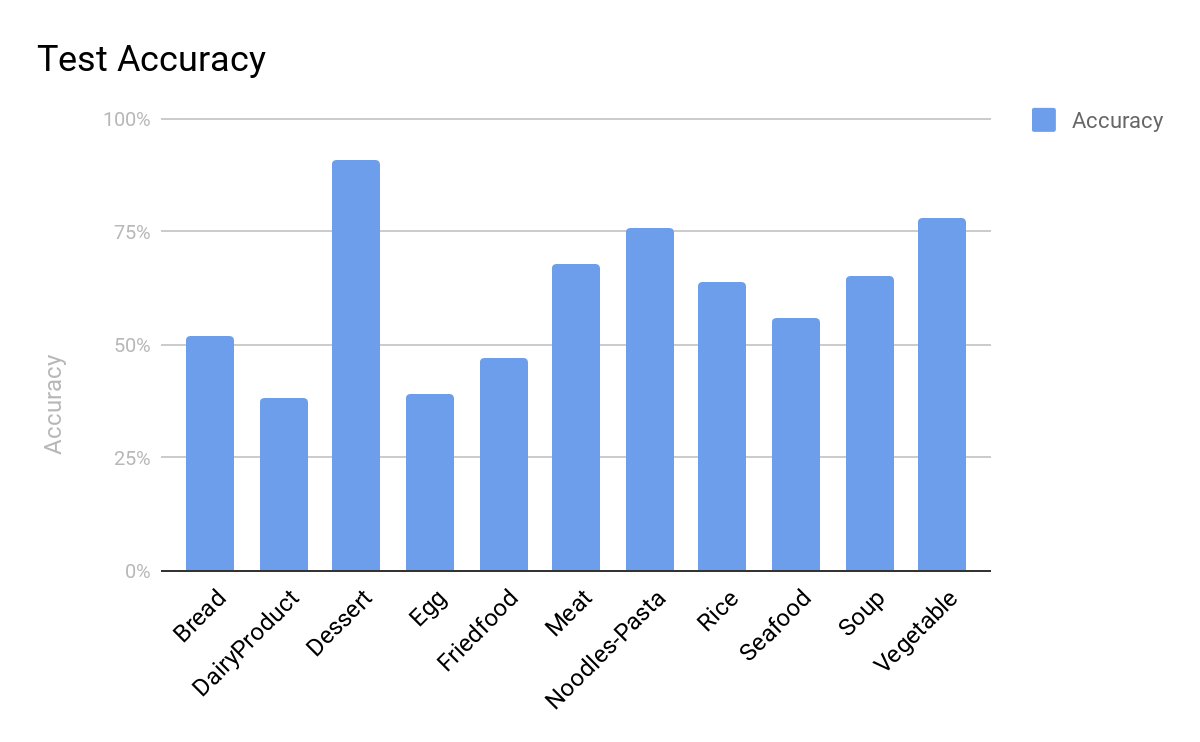
]),

}

### Training & Validation accuracy and learning rate decay



### Test **accuracy**

Accuracy of the network on the 3347 test images: 62.20%, and loss is: 0.025  


With the comparison of validation and test accuracy (~60%) with training accuracy (>99%), it’s obvious that the network is overfitting.

## With training data augmentation

### Source code of data transformation

data\_transforms = {

'train': transforms.Compose([

transforms.RandomResizedCrop(224),

transforms.RandomHorizontalFlip(),

transforms.ToTensor(),

transforms.Normalize([0.5548, 0.4508, 0.3435], [0.2281, 0.2384, 0.2376])

]),

'val': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

'test': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

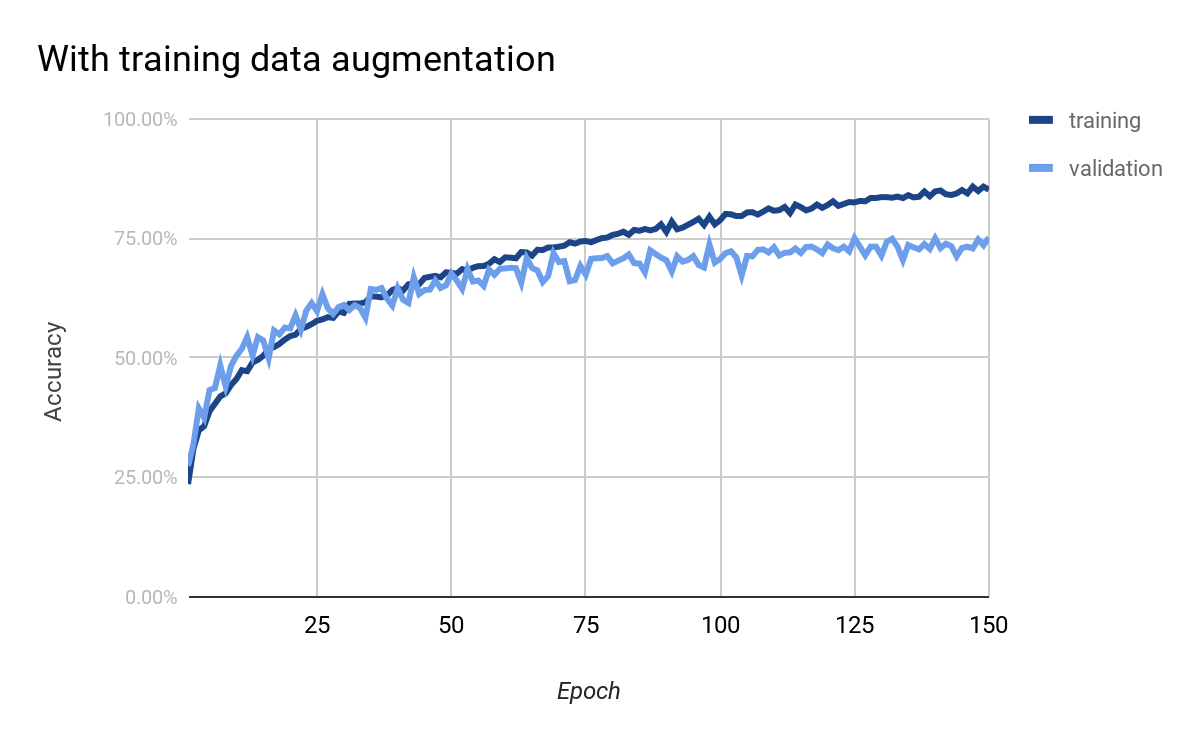
transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

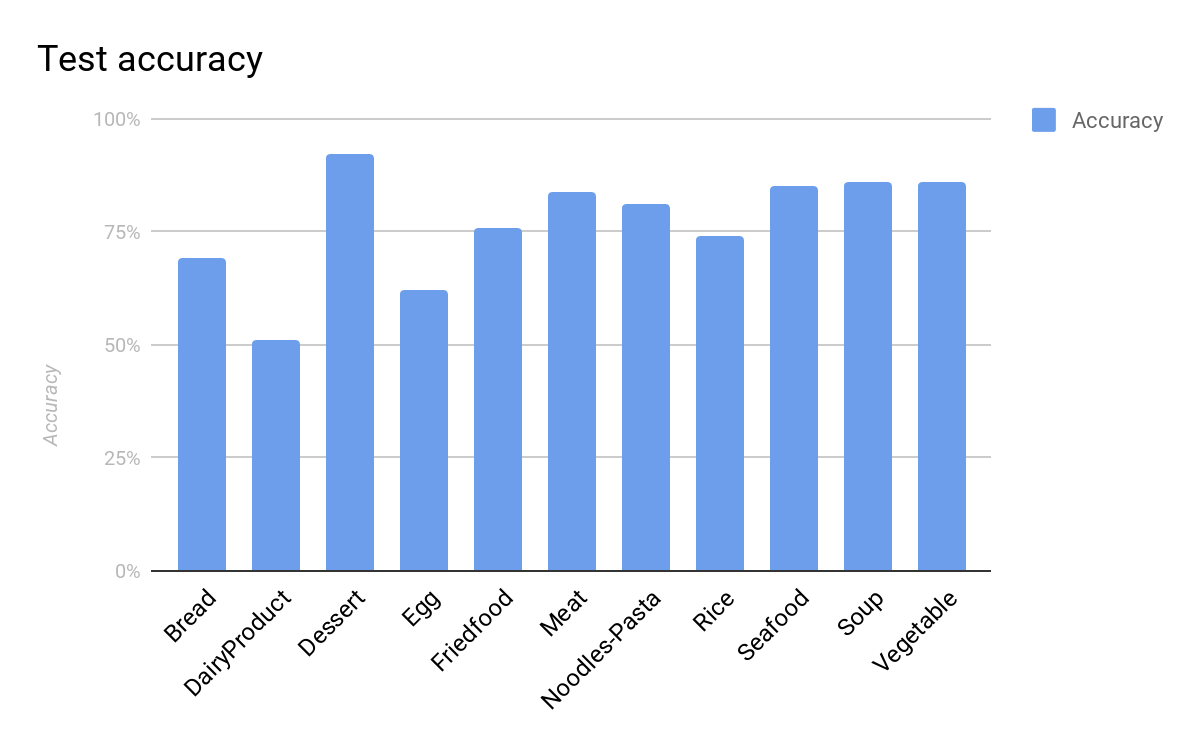
}

### Training & Validation accuracy and learning rate decay

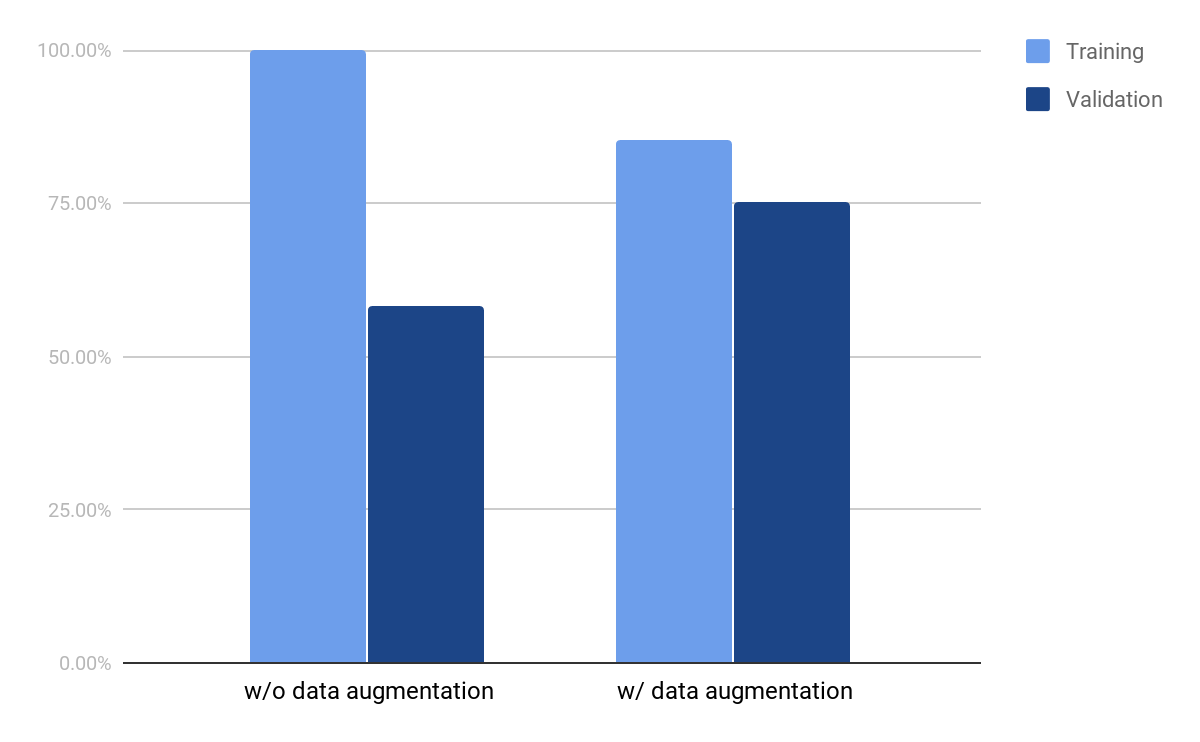


### Test accuracy

Accuracy of the network on the 3347 test images: 77.47%, and loss is: 0.013



By just using simple data augmentation of RandomResizedCrop and RandomResizedCrop, the differences of these two testing result are significant from 62% to 77%. The gap of training accuracy and validation accuracy also reduced from 42% down to 9%.



# LAB 2-2 : Use data augmentation to increase accuracy up to 87%

## Without training data augmentation

### Source code of data transformation

data\_transforms = {

'train': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

transforms.ToTensor(),

transforms.Normalize([0.5548, 0.4508, 0.3435], [0.2281, 0.2384, 0.2376])

]),

'val': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

'test': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

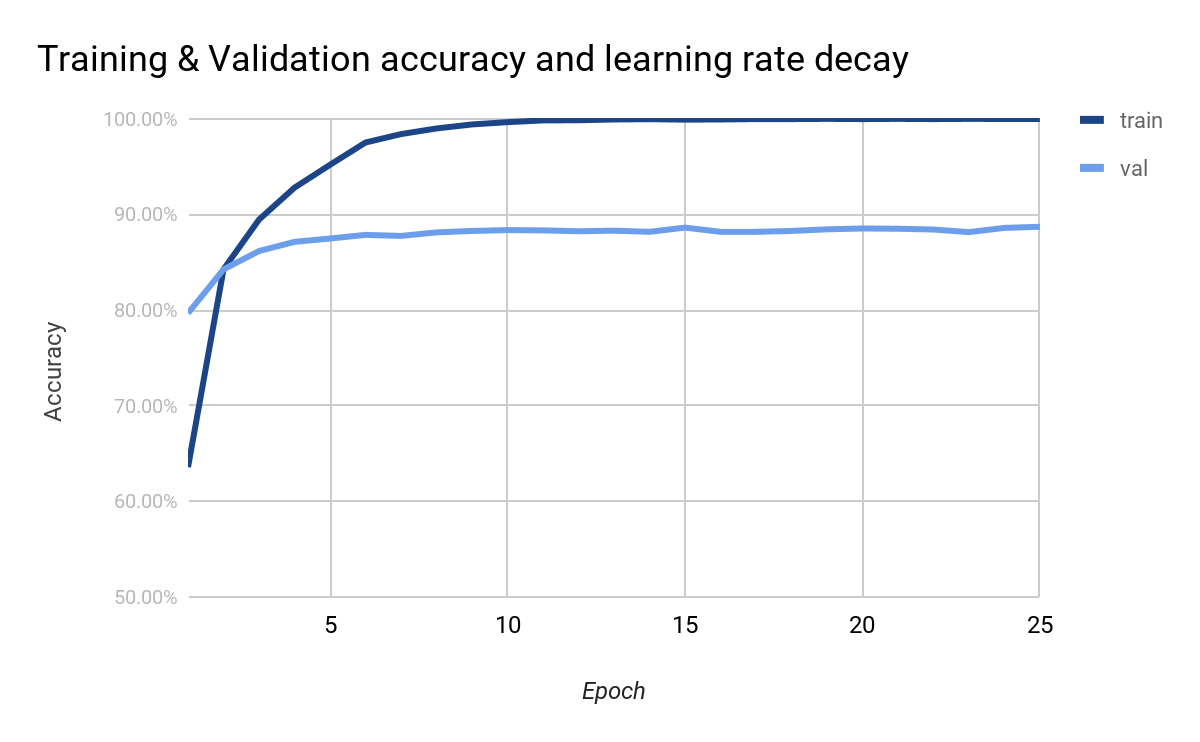
transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

}

### Training & Validation accuracy and learning rate decay



### Test accuracy

test2:

Accuracy of the network on the 838 test images: 91.29%, and loss is: 0.004

test2-aug1:

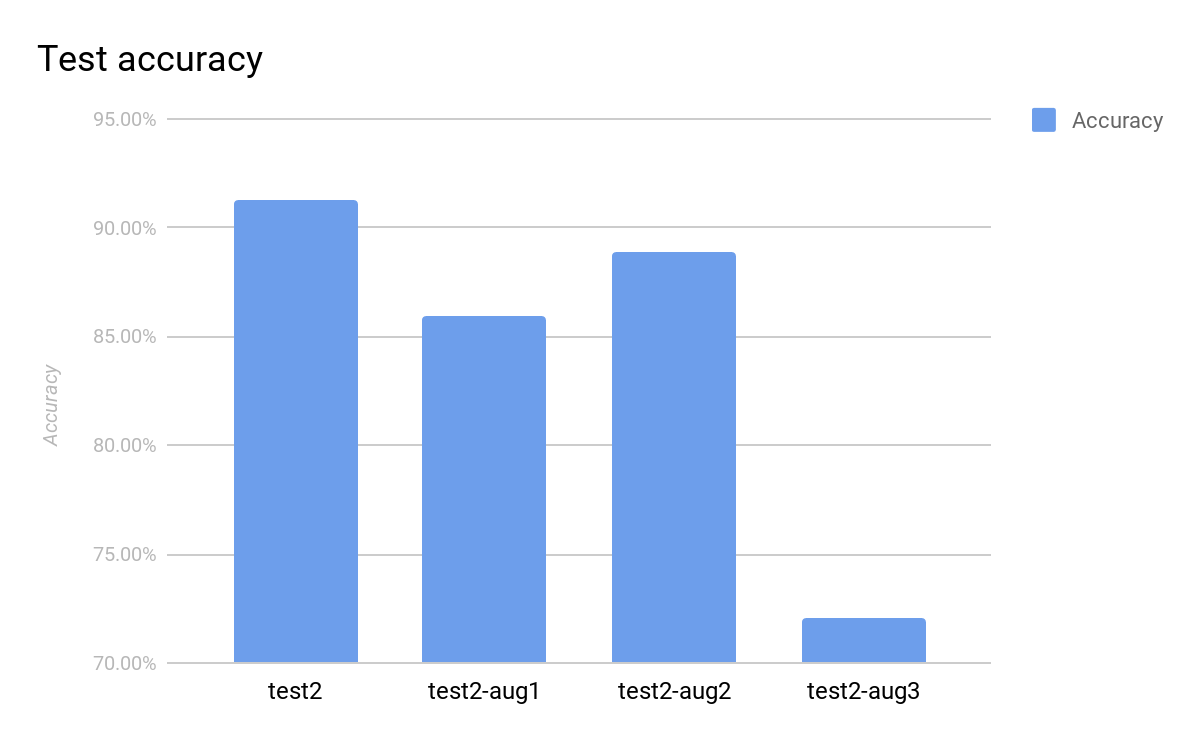
Accuracy of the network on the 838 test images: 85.92%, and loss is: 0.009

test2-aug2:

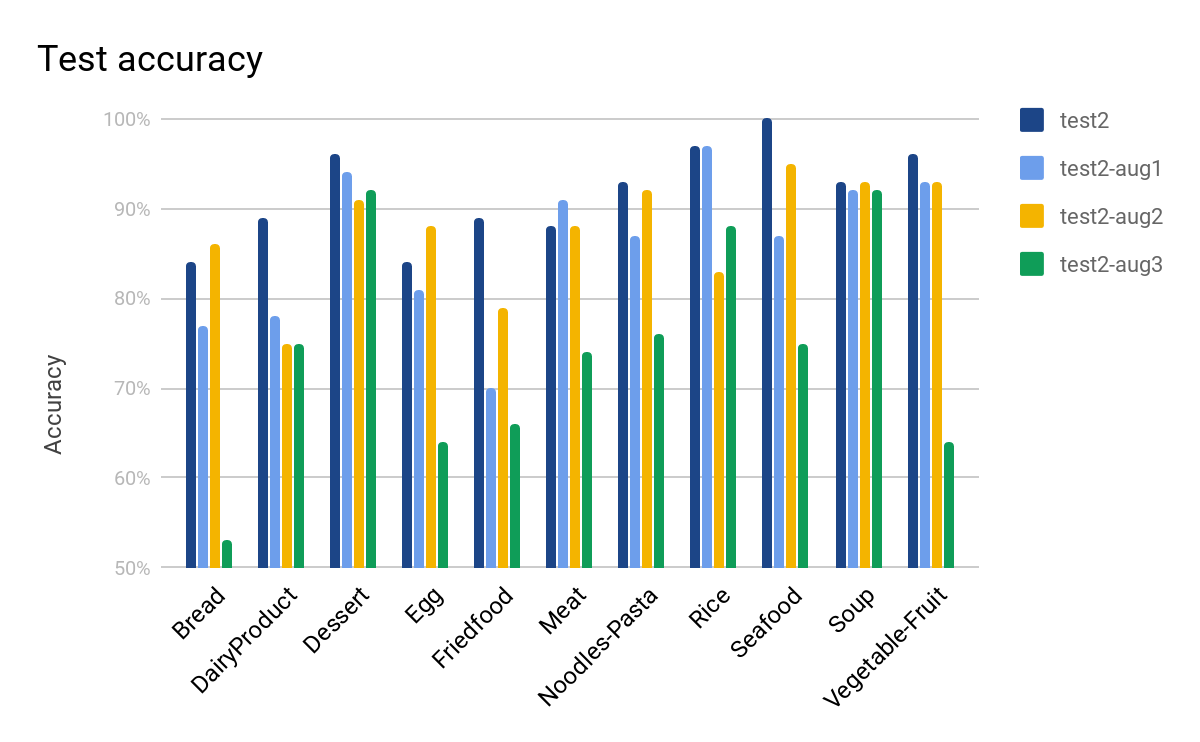
Accuracy of the network on the 838 test images: 88.90%, and loss is: 0.006

test2-aug3:

Accuracy of the network on the 833 test images: 72.03%, and loss is: 0.015



### Test accuracy in each class



## Data augmentation to increase the accuracy by Imgaug

### Source code of data transformation

class ImgAugTransform:

def \_\_init\_\_(self):

self.aug = iaa.Sequential([

iaa.Scale((224, 224)),

iaa.Sometimes(0.25, iaa.GaussianBlur(sigma=(0, 3.0))),

iaa.Fliplr(0.5),

iaa.Affine(rotate=(-20, 20), mode='symmetric'),

iaa.Sometimes(0.25,

iaa.OneOf([iaa.Dropout(p=(0, 0.1)),

iaa.CoarseDropout(0.1, size\_percent=0.5)])),

iaa.AddToHueAndSaturation(value=(-10, 10), per\_channel=True),

iaa.PiecewiseAffine(),

])

def \_\_call\_\_(self, img):

img = np.array(img)

return self.aug.augment\_image(img)

data\_transforms = {

'train': transforms.Compose([

ImgAugTransform(),

lambda x: Image.fromarray(x),

transforms.ToTensor(),

transforms.Normalize([0.5548, 0.4508, 0.3435], [0.2281, 0.2384, 0.2376])

]),

'val': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

test\_data: transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

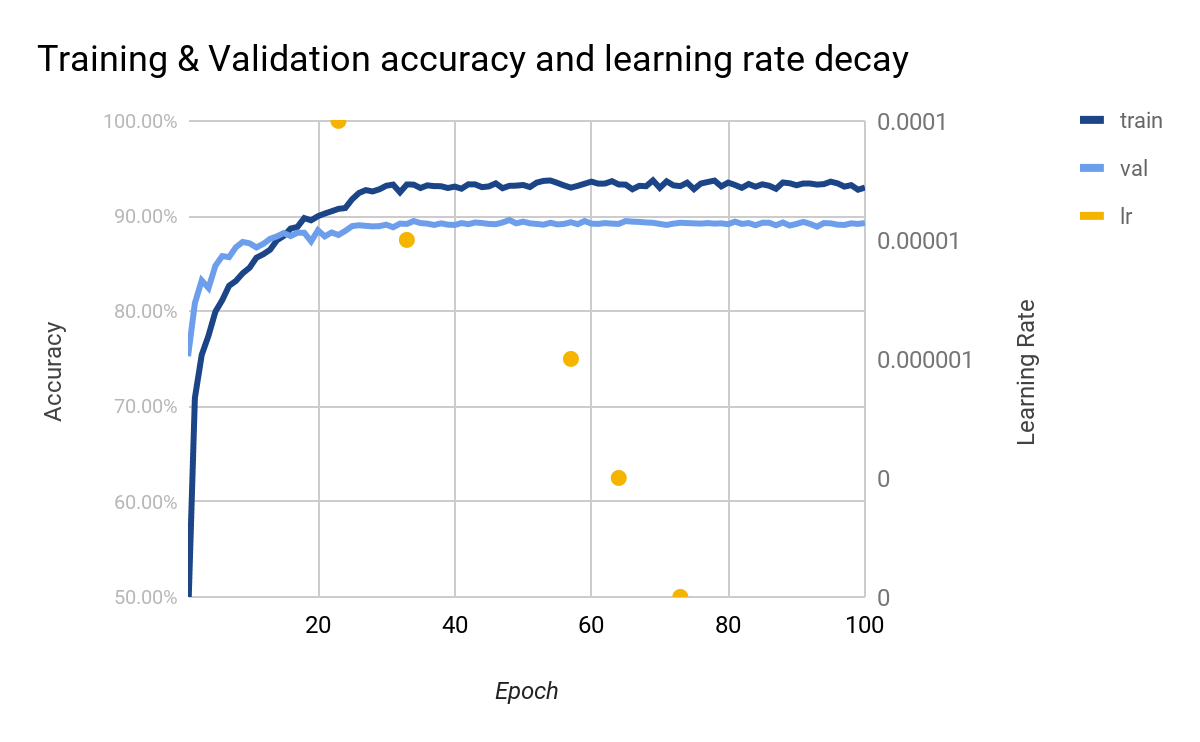
transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

}

### Training & Validation accuracy and learning rate decay



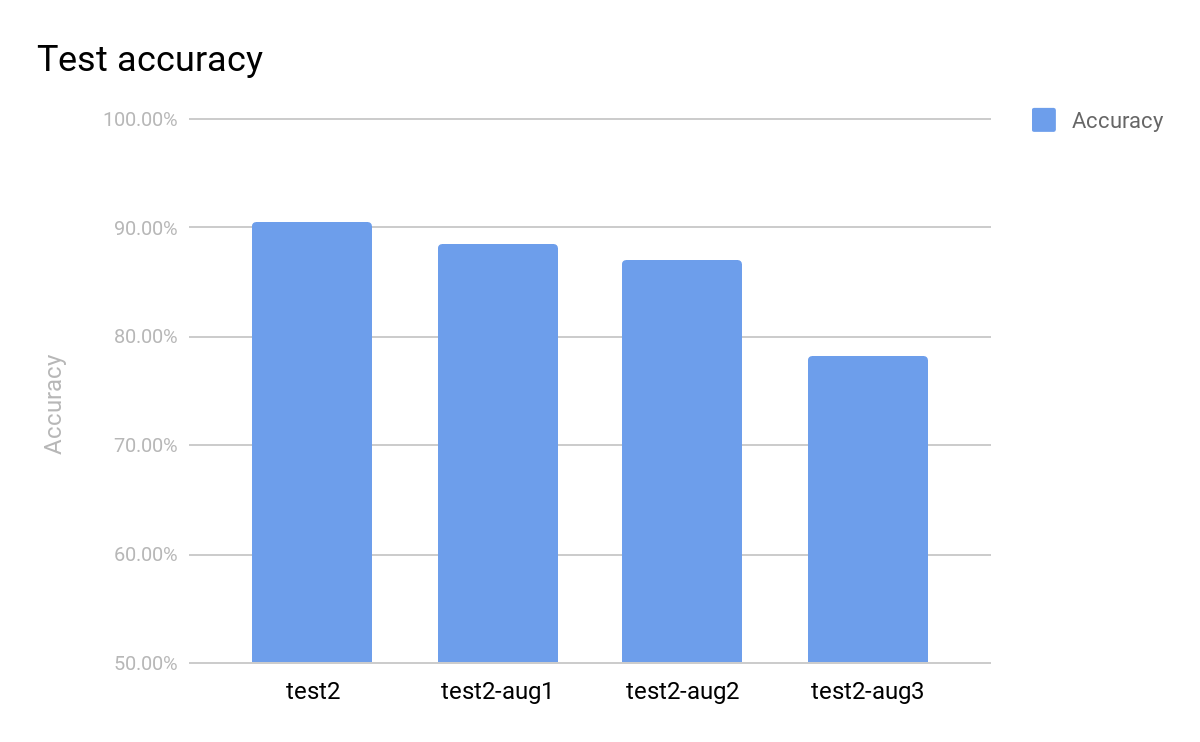
### Test accuracy

test2:

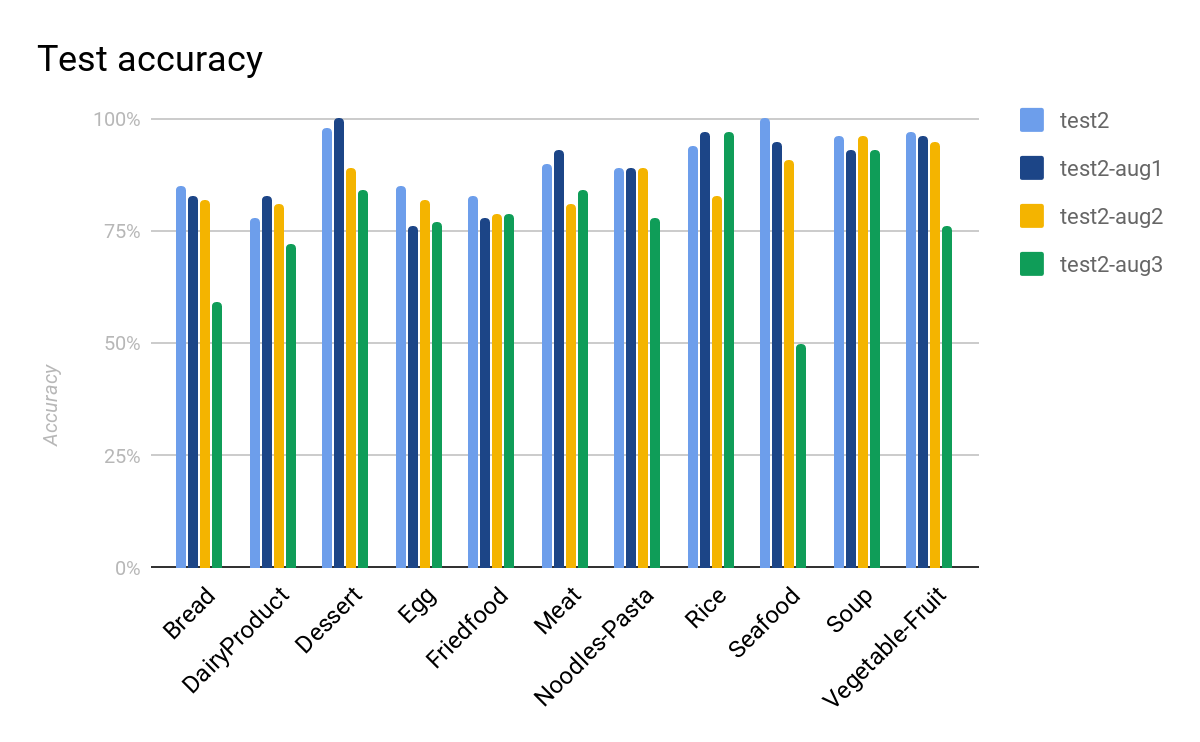
Accuracy of the network on the 838 test images: 90.45%, and loss is: 0.004  
test2-aug1:

Accuracy of the network on the 838 test images: 88.54%, and loss is: 0.006  
test2-aug2:

Accuracy of the network on the 838 test images: 86.99%, and loss is: 0.007  
test2-aug3:

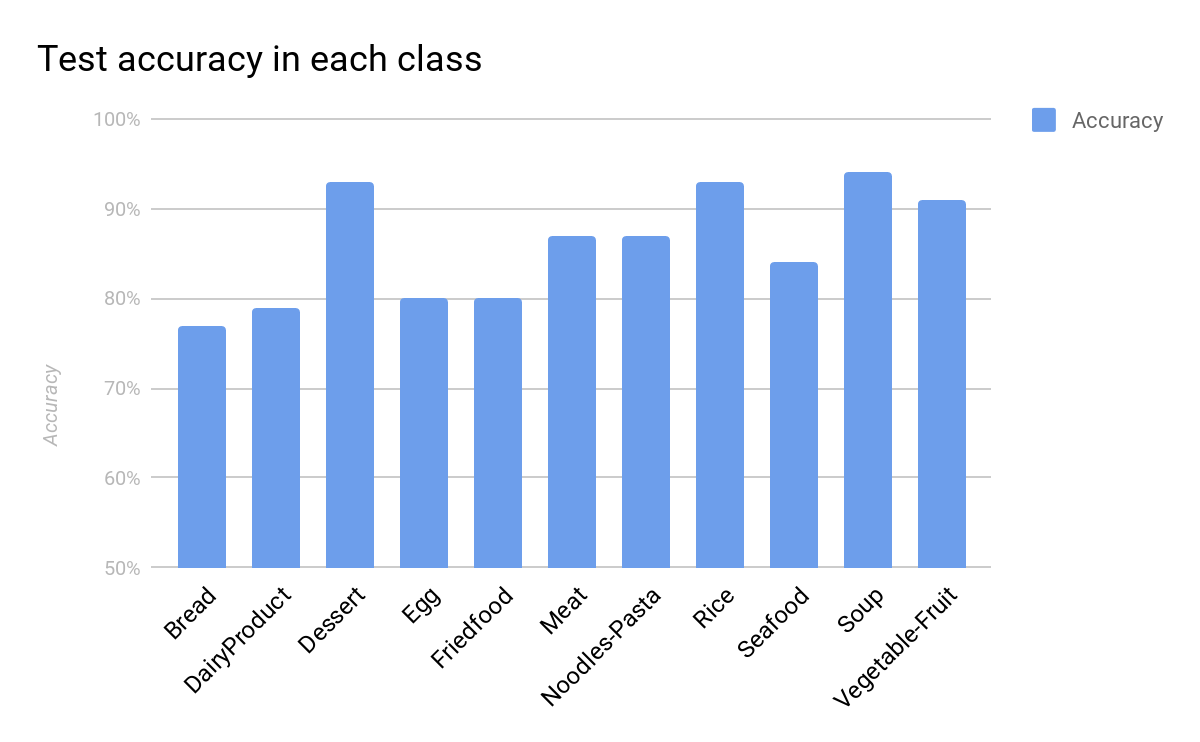
Accuracy of the network on the 833 test images: 78.15%, and loss is: 0.013  


### Test accuracy in each class under different data augmentation test cases



### Test accuracy in each class

Total test accuracy the network on the 3347 test images: 86.05%, and loss is: 0.007



## Data augmentation to increase the accuracy by Pytorch

### Source code of data transformation

data\_transforms = {

'train': transforms.Compose([

transforms.RandomHorizontalFlip(),

transforms.RandomVerticalFlip(),

transforms.RandomRotation((-180,180)),

transforms.RandomAffine(degrees=(-30,30), shear=(-20,20)),

transforms.RandomResizedCrop(224),

transforms.ToTensor(),

T ransforms.Normalize([0.5548, 0.4508, 0.3435], [0.2281, 0.2384, 0.2376])

]),

'val': transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

test\_data: transforms.Compose([

transforms.Resize(256),

transforms.CenterCrop(224),

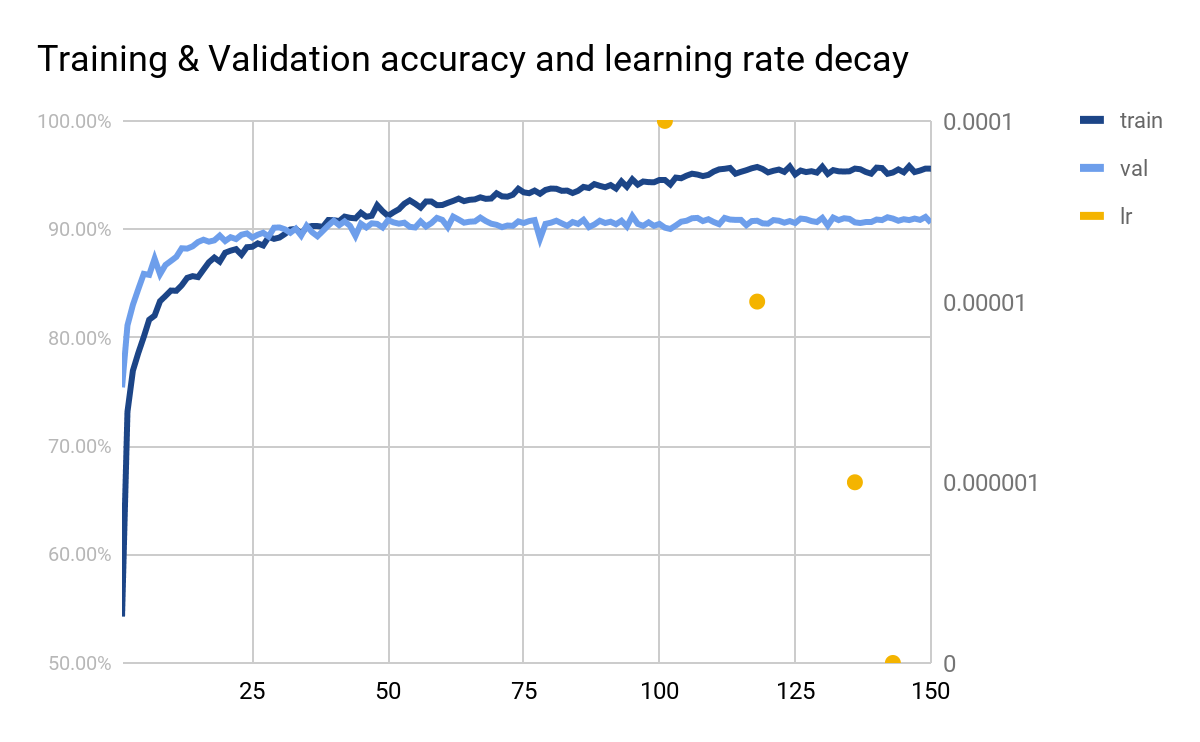
transforms.ToTensor(),

transforms.Normalize([0.5604, 0.4540, 0.3481], [0.2260, 0.2367, 0.2352])

]),

}

### Training & Validation accuracy and learning rate decay



### Test accuracy

test2:

Accuracy of the network on the 838 test images: 93.08%, and loss is: 0.004

test2-aug1:

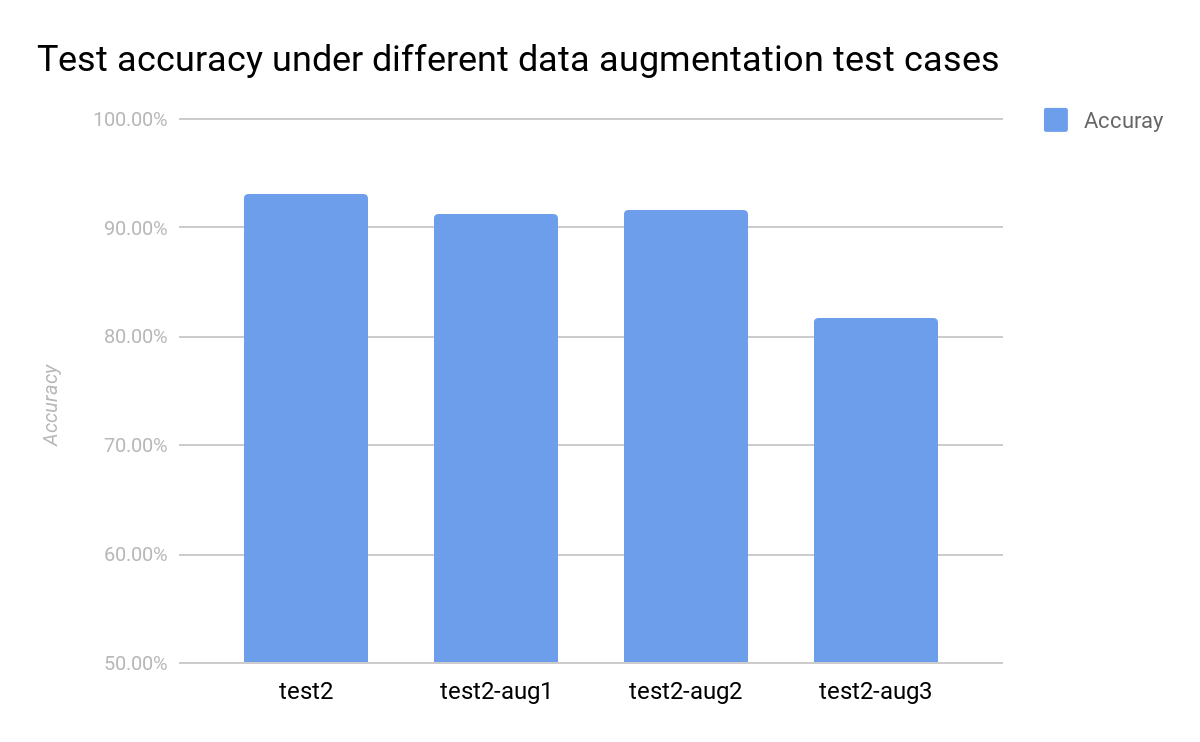
Accuracy of the network on the 838 test images: 91.29%, and loss is: 0.005

test2-aug2:

Accuracy of the network on the 838 test images: 91.65%, and loss is: 0.004

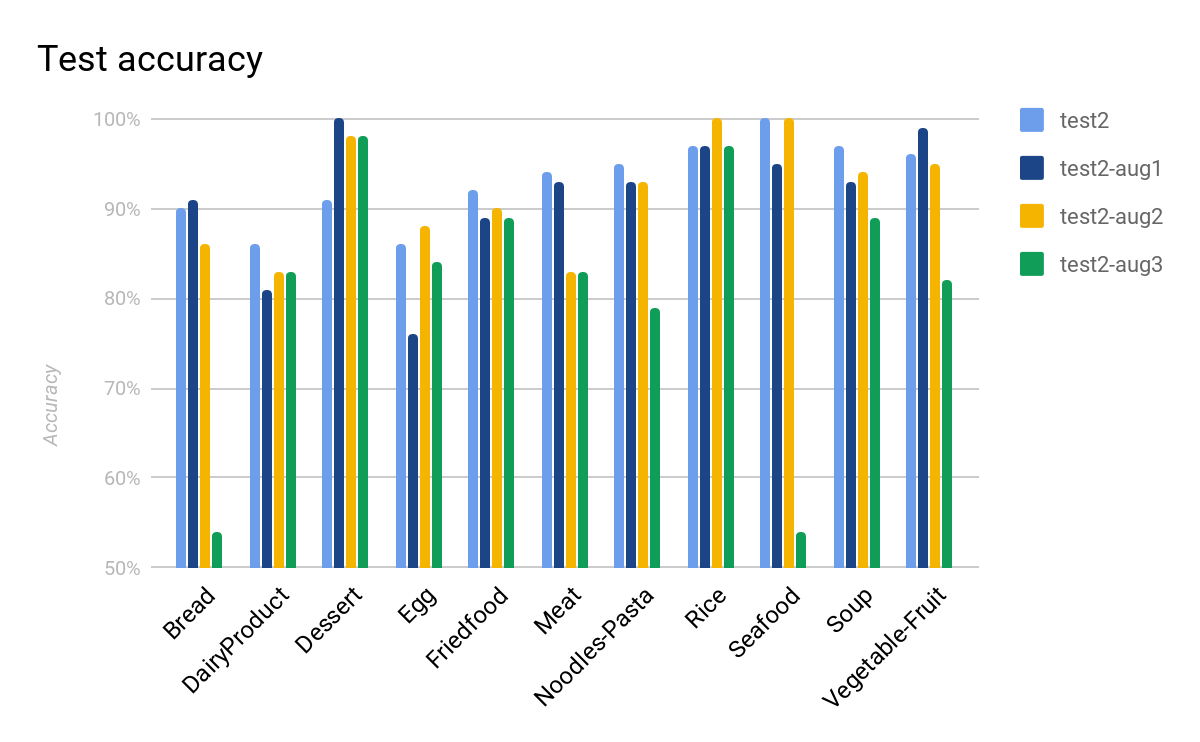
test2-aug3:

Accuracy of the network on the 833 test images: 81.63%, and loss is: 0.012



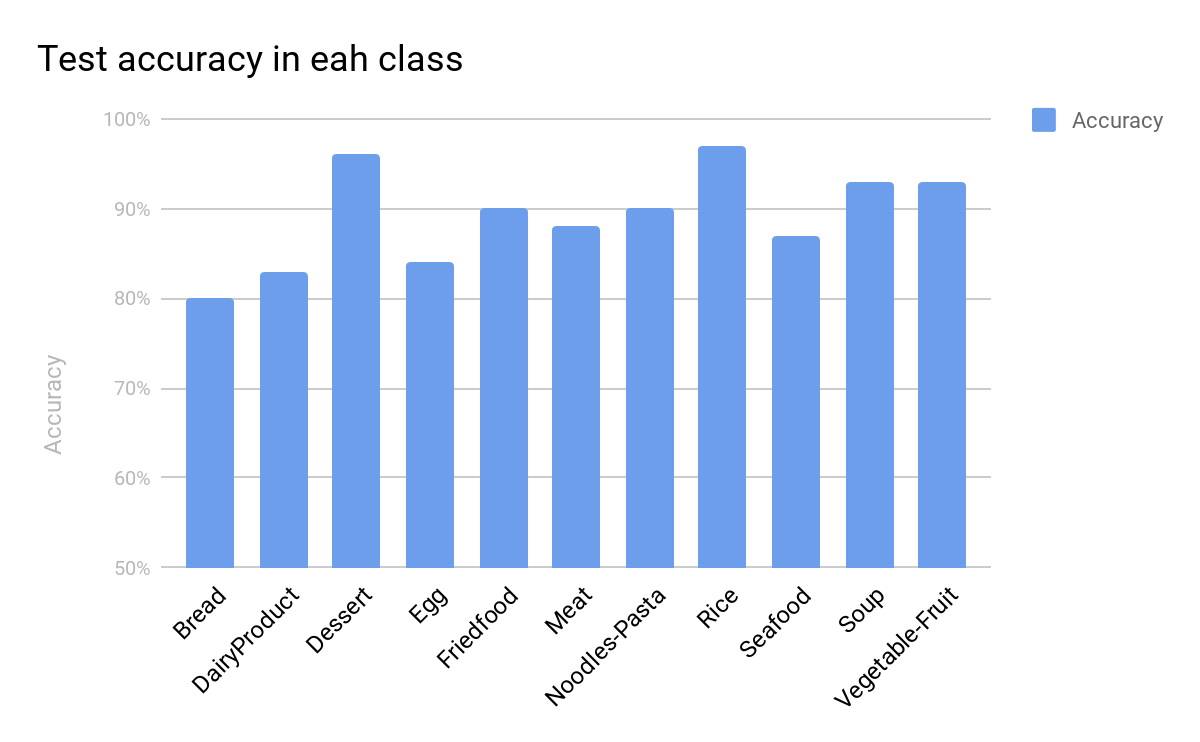
Total test accuracy the network on the 3347 test images: 89.42%, and loss is: 0.006

### Test accuracy in each class under different data augmentation test cases

****

### Test accuracy in each class

Total test accuracy the network on the 3347 test images: 89.42%, and loss is: 0.006

****

* Problems & solutions :
* Experiment setup :
* Results :
* Analysis :