Big cats classifier

Dataset

kaggle

10 Big Cats of the Wild - Image Classification

Used Technologies

colab

jupyter

PyTorch

Learning parameters

Parameters:

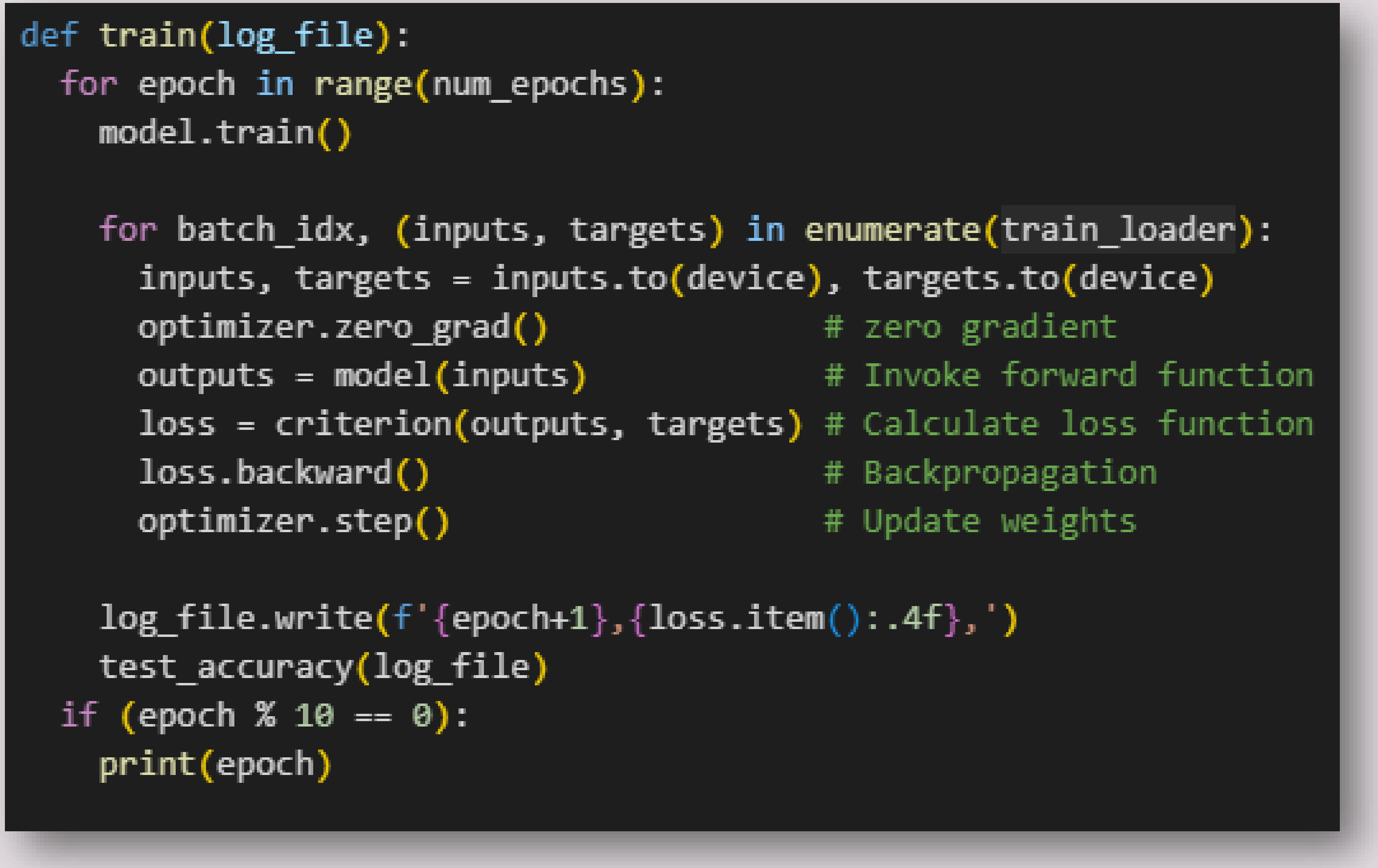
- Batch size
- Epochs
- Learning rate
- Criterion (CrossEntropyLoss)

Optimizers:

- Adam
- SGD (Stochastic Gradient Descent)

Data preparation

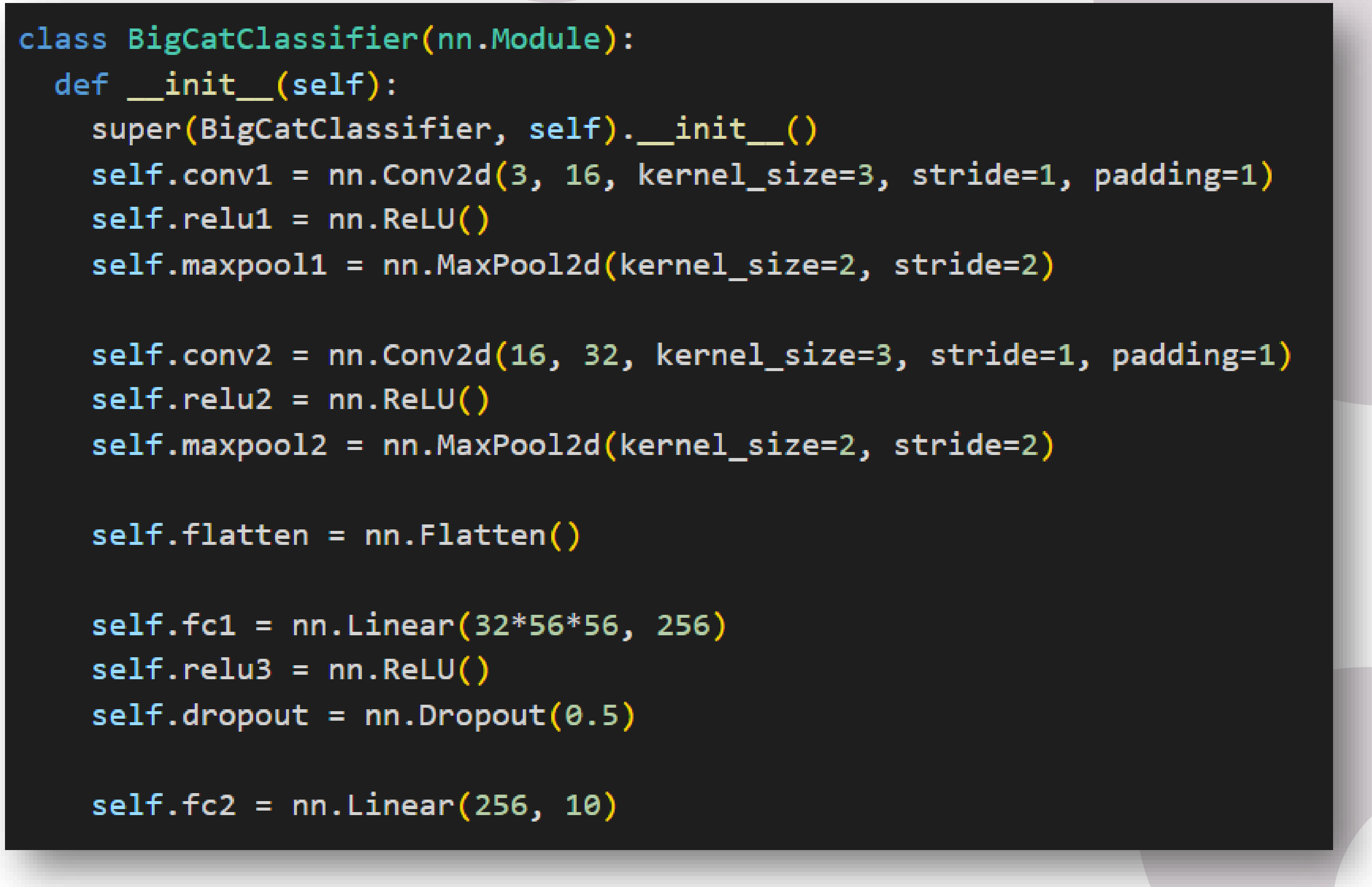
Training loop



```
def train(log_file):
  for epoch in range(num_epochs):
    model.train()

    for batch_idx, (inputs, targets) in enumerate(train_loader):
      inputs, targets = inputs.to(device), targets.to(device)
      optimizer.zero_grad()              # zero gradient
      outputs = model(inputs)            # Invoke forward function
      loss = criterion(outputs, targets) # Calculate loss function
      loss.backward()                    # Backpropagation
      optimizer.step()                   # Update weights

    log_file.write(f'{epoch+1},{loss.item():.4f},')
    test_accuracy(log_file)
  if (epoch % 10 == 0):
    print(epoch)
```

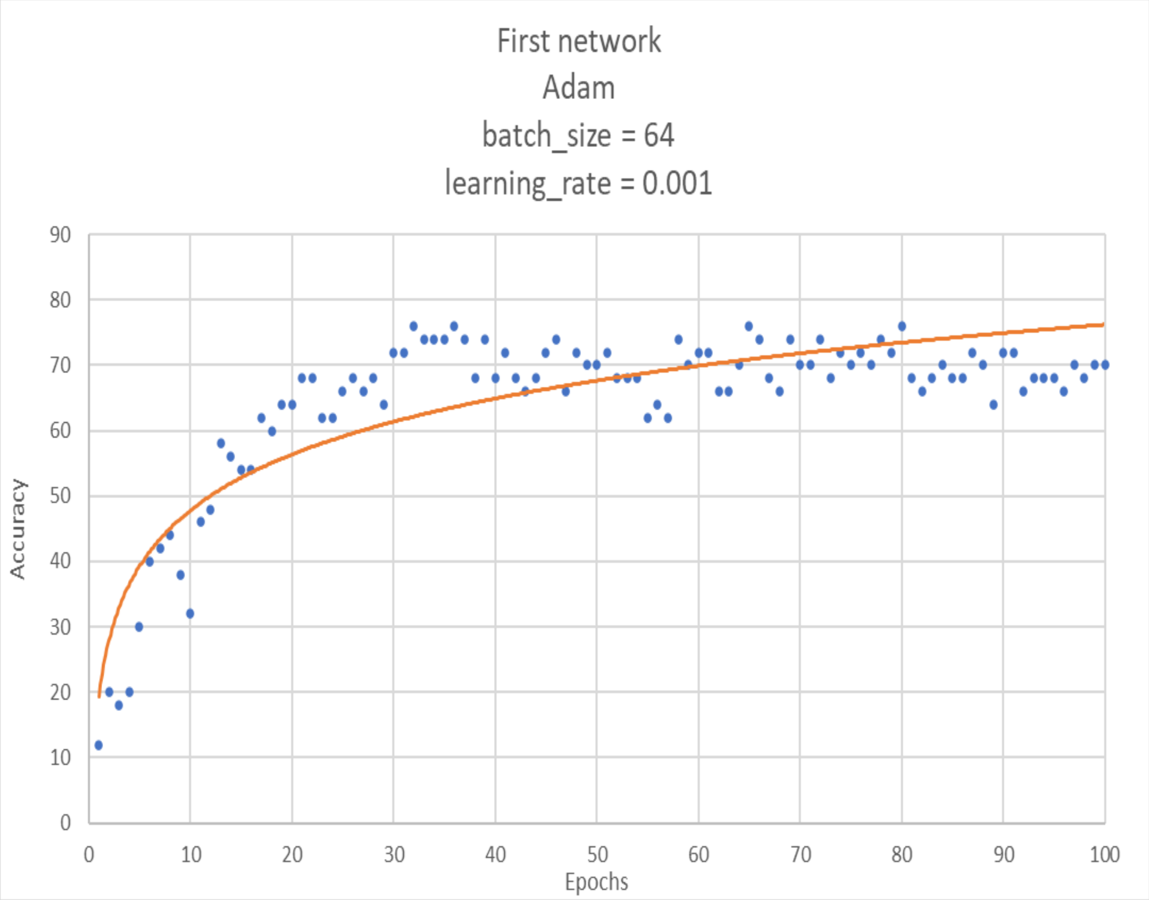
Neural network architecture

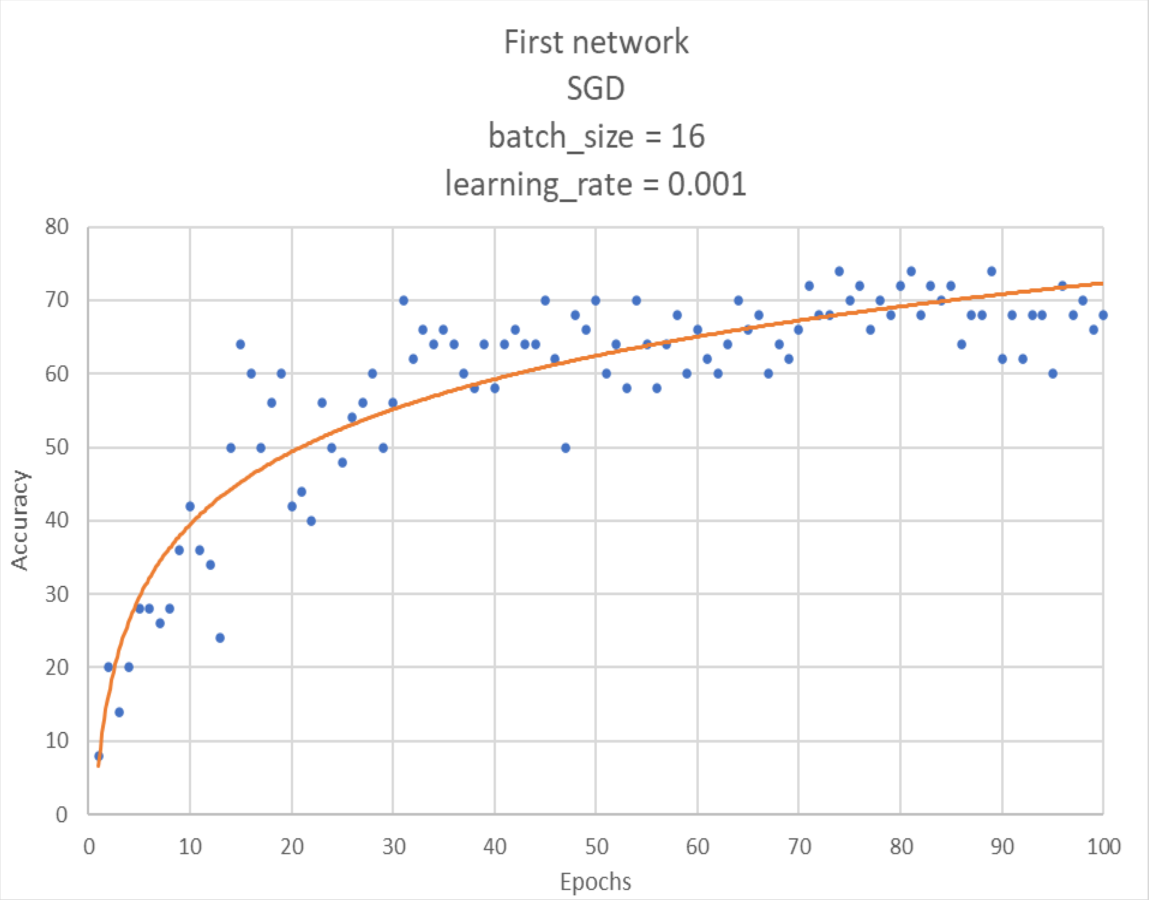
First network



```python
class BigCatClassifier(nn.Module):
  def __init__(self):
    super(BigCatClassifier, self).__init__()
    self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1)
    self.relu1 = nn.ReLU()
    self.maxpool1 = nn.MaxPool2d(kernel_size=2, stride=2)

    self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1)
    self.relu2 = nn.ReLU()
    self.maxpool2 = nn.MaxPool2d(kernel_size=2, stride=2)

    self.flatten = nn.Flatten()

    self.fc1 = nn.Linear(32*56*56, 256)
    self.relu3 = nn.ReLU()
    self.dropout = nn.Dropout(0.5)

    self.fc2 = nn.Linear(256, 10)
```

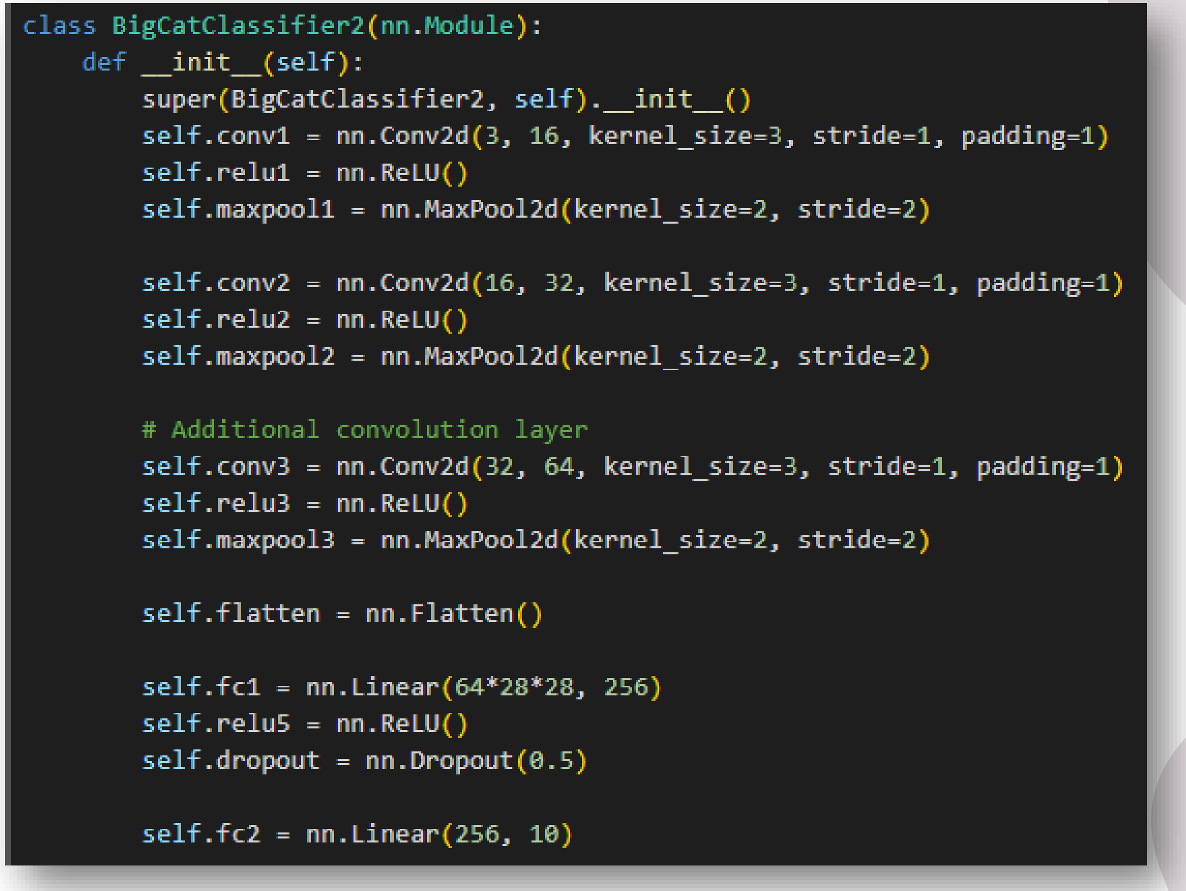
First net results





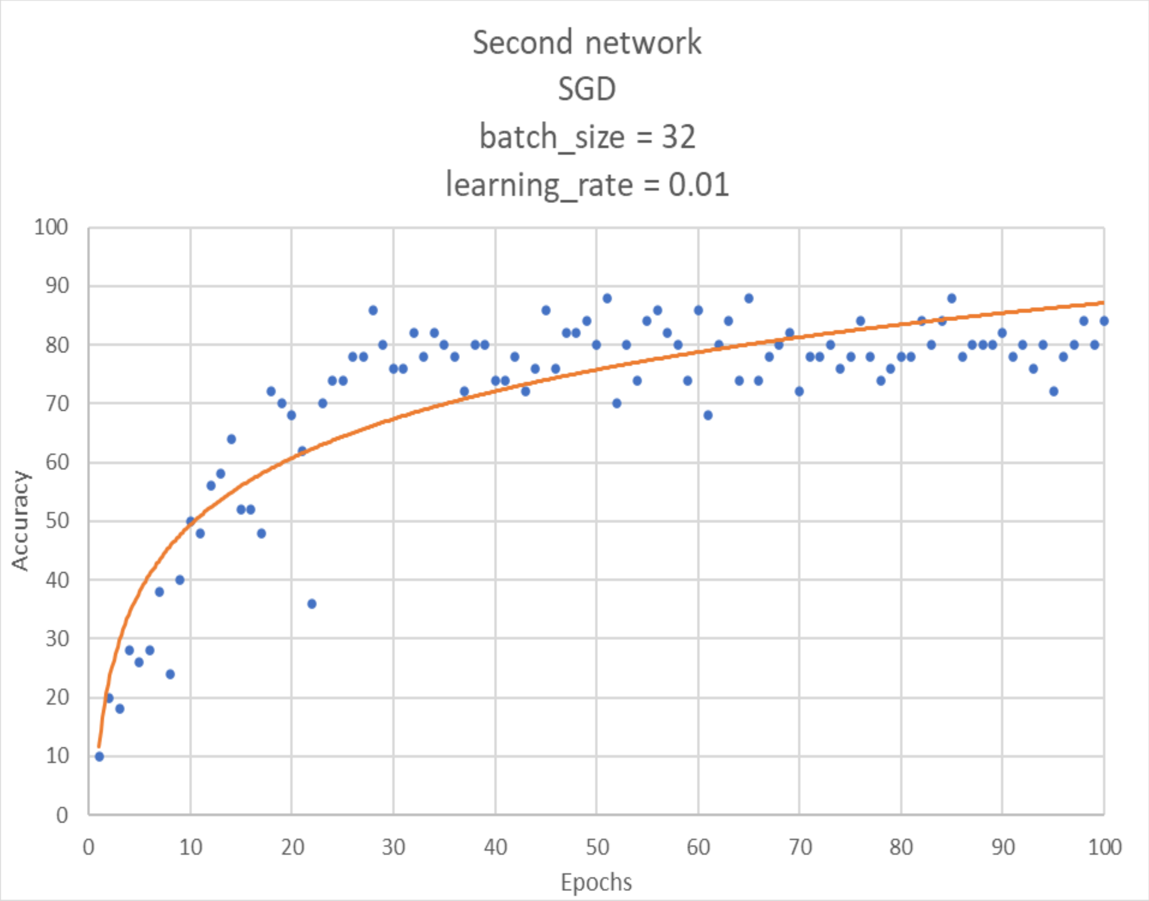
Neural network architecture

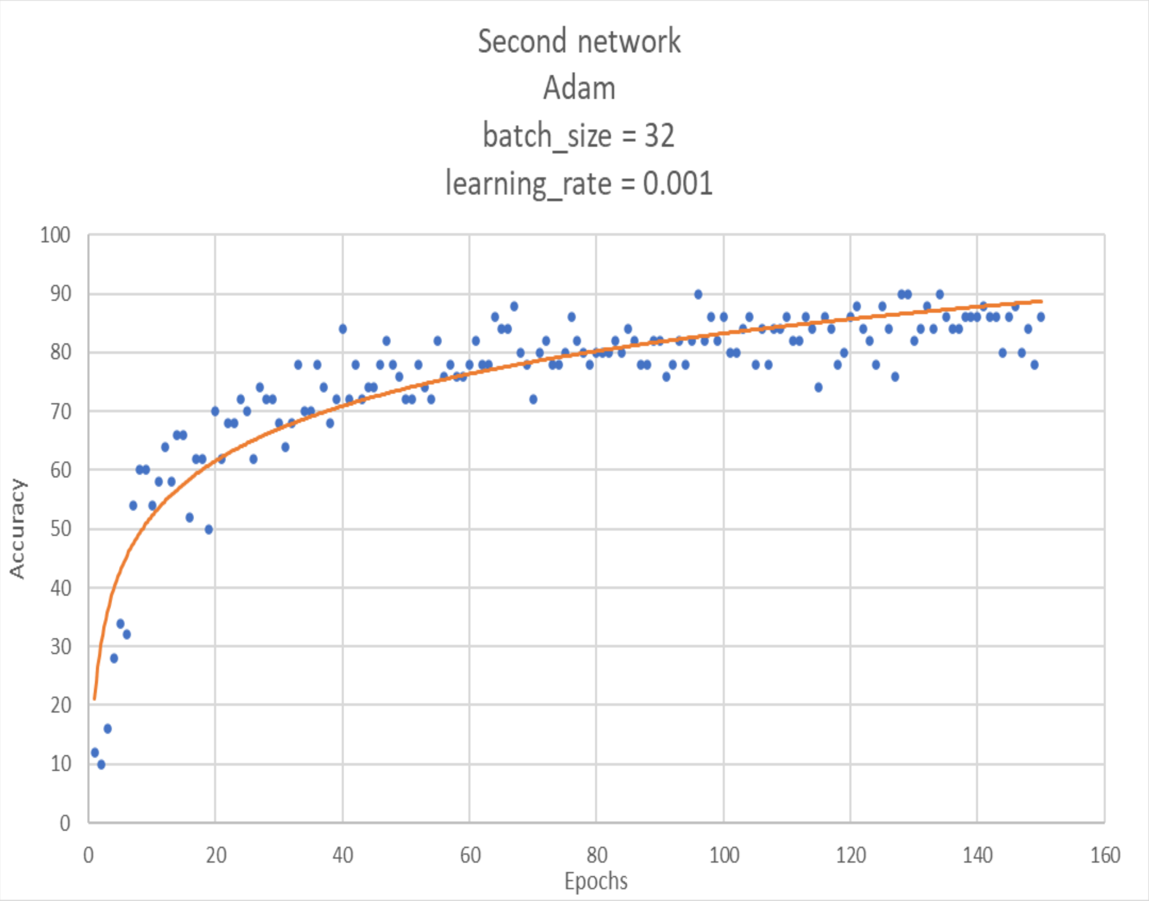
Second network



```python
class BigCatClassifier2(nn.Module):
    def __init__(self):
        super(BigCatClassifier2, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1)
        self.relu1 = nn.ReLU()
        self.maxpool1 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1)
        self.relu2 = nn.ReLU()
        self.maxpool2 = nn.MaxPool2d(kernel_size=2, stride=2)

        # Additional convolution layer
        self.conv3 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.relu3 = nn.ReLU()
        self.maxpool3 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.flatten = nn.Flatten()

        self.fc1 = nn.Linear(64*28*28, 256)
        self.relu5 = nn.ReLU()
        self.dropout = nn.Dropout(0.5)

        self.fc2 = nn.Linear(256, 10)
```

Second net results





Next steps

Thanks for listening