Neural Architecture Search with Reinforcement Learning

Kesanupalli Srivatsava MT18054
Vasisht Duddu 2015137

Source Code

https://github.com/srivatsavak/RL_Project

Used utils.py from Google AutoML for environment: Provides the state space format and RNN architecture

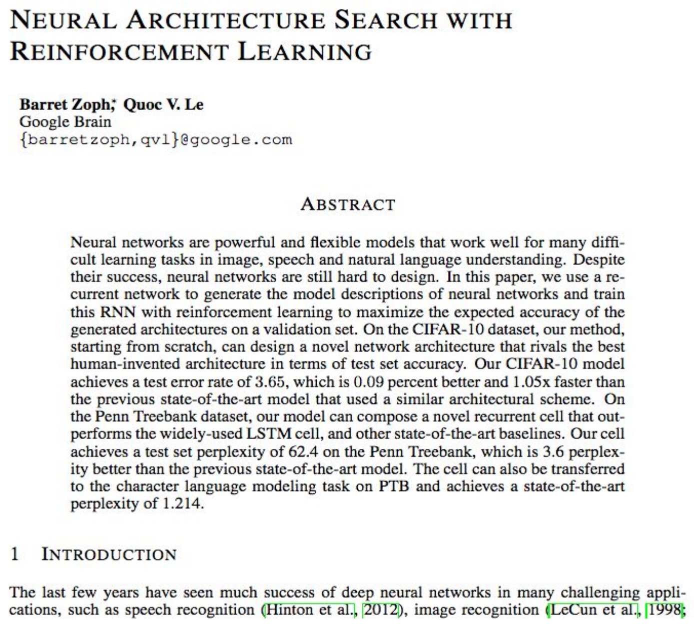
Goal of the Project

- Neural Architectures Search using Reinforce Algorithm
    - Compare Baseline and Without Baseline Reinforce
    - Verify paper’s claim that baseline result for reinforce has lower variance

- Implement the paper on FashionMNIST dataset
    - Did not compare with paper results with CIFAR10
    - requires more computational resources: 400 GPU days and larger architectures

Referenced Papers

Neural Architecture Search with Reinforcement Learning. ICLR 2017.
https://arxiv.org/abs/1611.01578



NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING

Barret Zoph*, **Quoc V. Le**
Google Brain
{barretzoph,qvl}@google.com

ABSTRACT

Neural networks are powerful and flexible models that work well for many difficult learning tasks in image, speech and natural language understanding. Despite their success, neural networks are still hard to design. In this paper, we use a recurrent network to generate the model descriptions of neural networks and train this RNN with reinforcement learning to maximize the expected accuracy of the generated architectures on a validation set. On the CIFAR-10 dataset, our method, starting from scratch, can design a novel network architecture that rivals the best human-invented architecture in terms of test set accuracy. Our CIFAR-10 model achieves a test error rate of 3.65, which is 0.09 percent better and 1.05x faster than the previous state-of-the-art model that used a similar architectural scheme. On the Penn Treebank dataset, our model can compose a novel recurrent cell that outperforms the widely-used LSTM cell, and other state-of-the-art baselines. Our cell achieves a test set perplexity of 62.4 on the Penn Treebank, which is 3.6 perplexity better than the previous state-of-the-art model. The cell can also be transferred to the character language modeling task on PTB and achieves a state-of-the-art perplexity of 1.214.

1 INTRODUCTION

The last few years have seen much success of deep neural networks in many challenging applications, such as speech recognition (Hinton et al., 2012), image recognition (LeCun et al., 1998;

Environment, Agents, Rewards, and etc.

- Search Space
  - Different Convolutional Models with fixed depth
- Agent
  - RNN Controller to output different hyperparameters: stride; filter size; kernels
- Reward
  - Considered reward clipping [-0.05,0.05] or [-0.1,0.1]
  - Computed using (Accuracy - Moving Accuracy)
- Performance Metric
  - Validation Accuracy

RL Algorithm: REINFORCE with baseline

Track moving accuracy and use them as a baseline

```
self.moving_acc = self.beta * self.moving_acc + (1 - self.beta) * acc
```

Compute Reward as a difference between current accuracy and moving accuracy

```
reward = (acc - self.moving_acc)
```

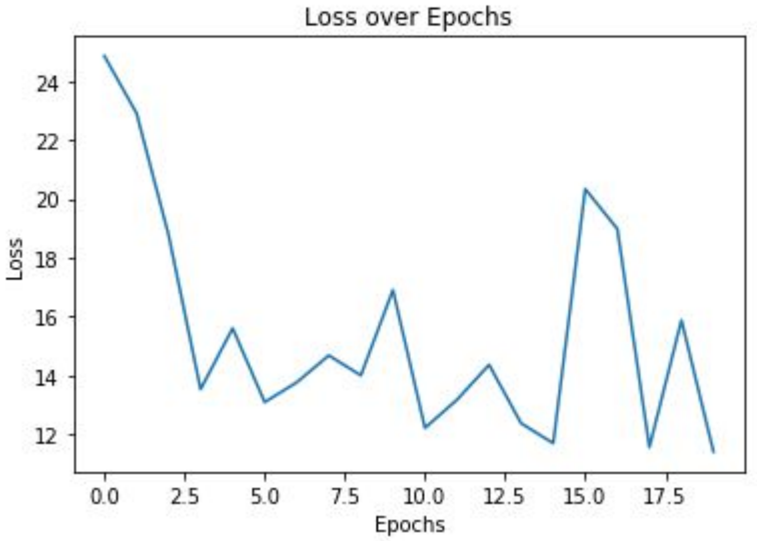
Clip the rewards to lie in [-0.1,0.1]

```
reward = np.clip(reward, -0.1, 0.1)
```

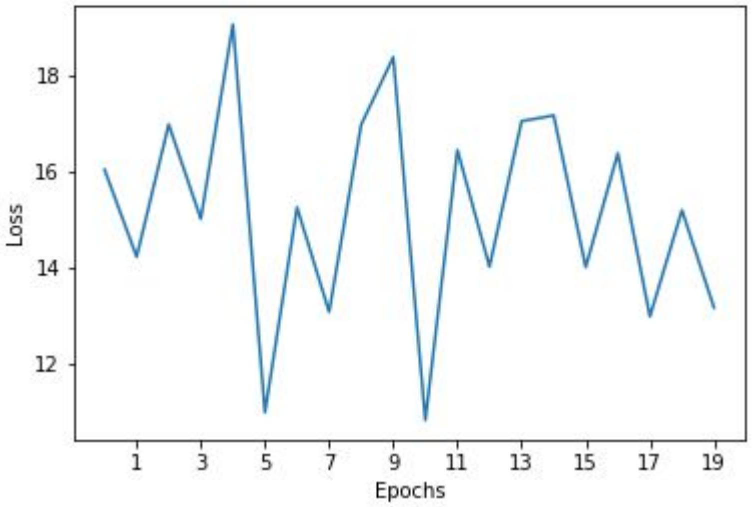
Use the reward and accuracy to compute loss

```
self.total_loss = policy_gradient_loss + self.hyper * reg_loss
```

Results - 1 :: Training Curves

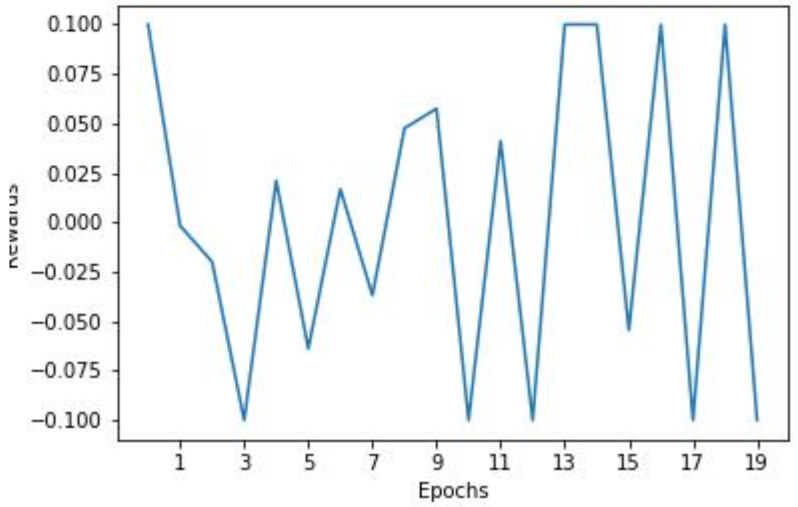


Baseline

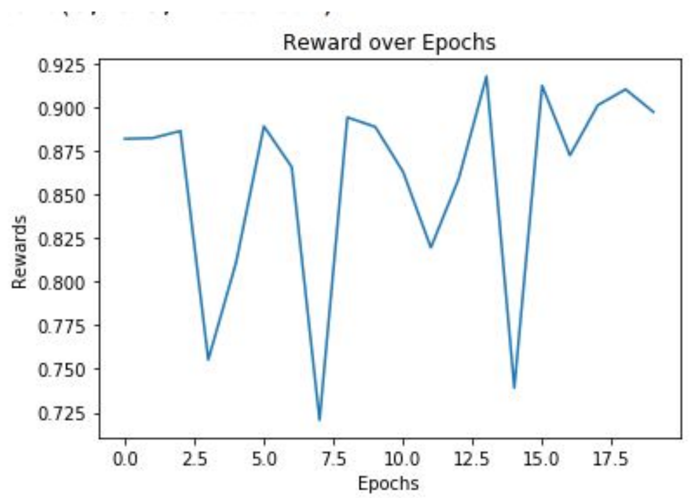


No Baseline

Results - 2 :: Average Reward



Baseline (Clipped Rewards)



No Baseline

Results - 3 :: Std. Deviation and Variance

Without Baseline Rewards:

Std. Deviation: 0.764

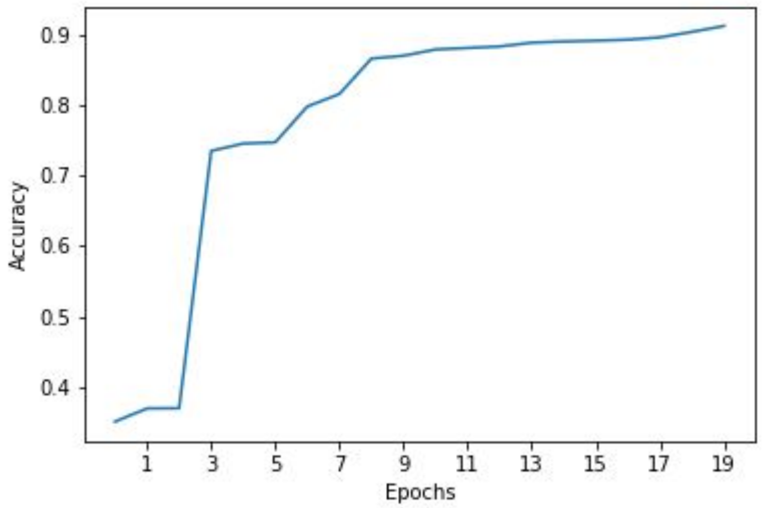
Variance: 0.0059

Baseline Rewards:

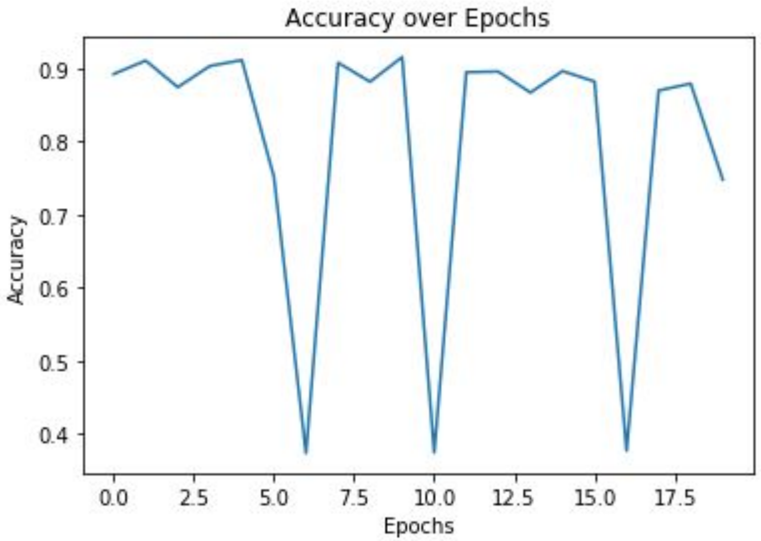
Std. Deviation: 0.0587

Variance: 0.003454

Results - 4 :: Accuracy over time



Baseline



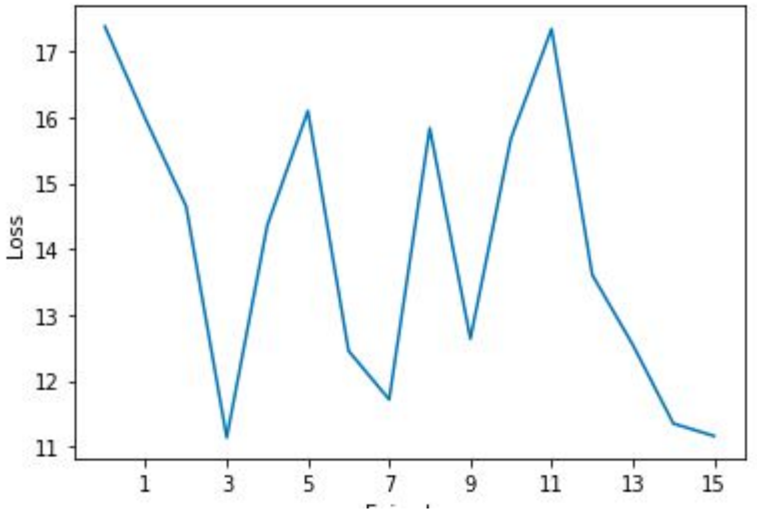
No Baseline

RL Algorithm :: Actor Critic

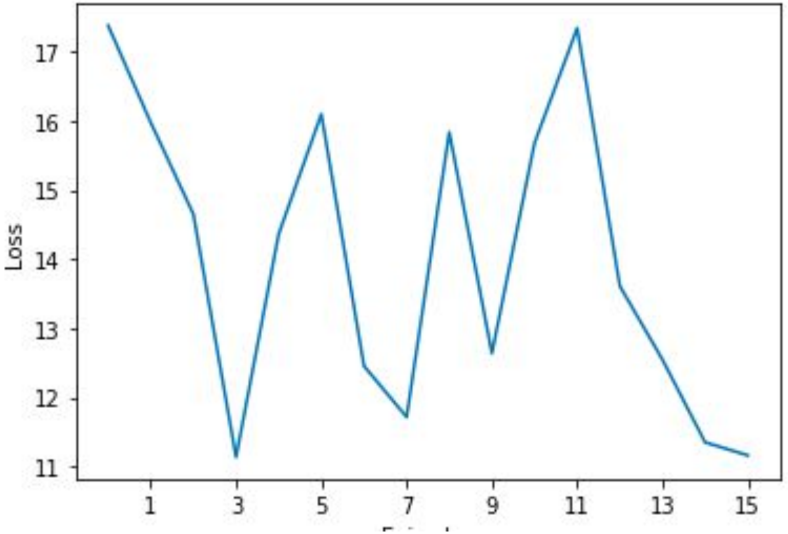
Here we predict the next architecture and use it’s accuracy to update the current model’s parameters

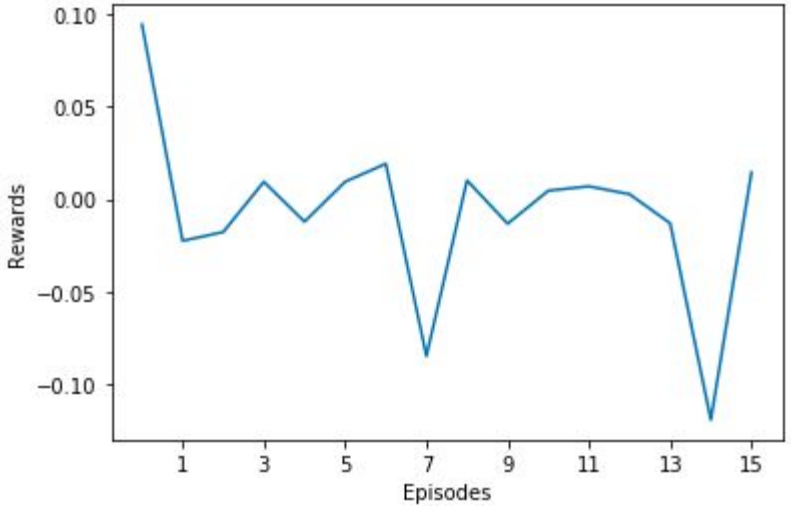
```
next_reward, next_acc = manager.get_rewards(model_fn,state_space.parse_state_space_list(temp_action))

reward+=next_re
```



Loss and Rewards

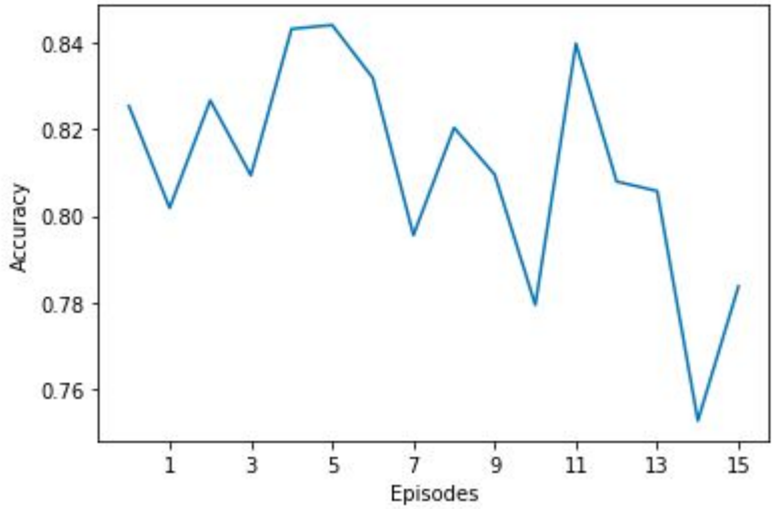




Standard Deviation for Rewards: `0.044685700281093185`

Variance for Rewards: `0.0019968118096116917`

Accuracy


Accuracy
Episodes