Neural Architecture Search using Reinforcement Learning

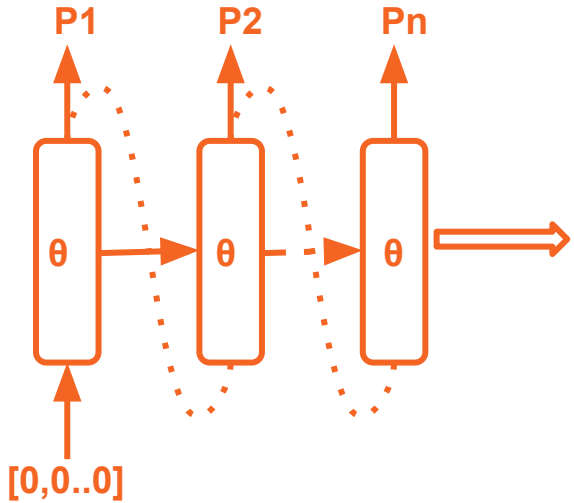
Dhruv Ramani

Compute Gradients for LSTM using REINFORCE

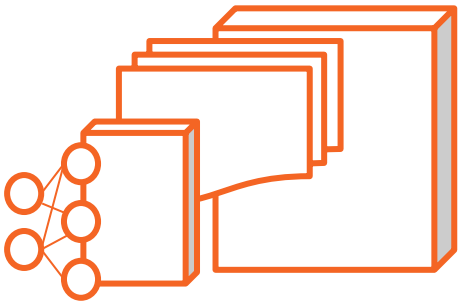
$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_{\theta_c}\log P(a_t|a_{(t-1):1};\theta_c)R_k$$

Reward : Validation Accuracy

Train Model



Construct Model using Generated Hyper Parameters



Single Conv, FC Layer

Code (Summary)

def neural_search(self):
LSTM -> All outputs and the final outputs

def gen_hyperparams(self, outputs):
Parse Outputs to get Hyperparameters

def construct_model(self, hyperparams):
Constructs the model based on generated hyperparameters

def model_loss(self, logits, labels):
return *tf.reduce_mean(tf.softmax_cross_entropy_with_logits(logits, labels))*

def train_model(self, loss):
Minimize *model_loss*

def reinforce(self, prob):
return *tf.reduce_mean(tf.log(prob))*

def train_controller(self, reinforce_loss, val_accuracy):
Take Gradients out -> Multiply with *val_accuracy* -> Apply Gradients