Anyone working on deep learning capstone project?

pkpp1233 **Jul 14**

It seems like the discussion forum is pretty silent when it comes to the Deep Learning capstone project.

Anyone working on it and want to collaborate / troubleshoot together?

herchu **Jul 15**

I noticed that too. I haven't seen any mention of the project in the forum.

Personally I have it in my pipeline. Right now I'm involved in a NLP project unrelated to TensorFlow and don't have much time to invest there. But I reckon that if keep postponing it I'll forget what I learnt of TF and getting back to it would be tough.

bri **Jul 21**

I'm working on this project and would like to have someone to troubleshoot with.

bri **Jul 21**

I watched the videos for Stanford CS231n and found them very helpful in understanding the concepts.
https://www.youtube.com/playlist?list=PLwQyV9I_3POsyBPRNUU_ryNfXzgfkiw2p

brian_20792649369148 **Jul 21**

I am currently working on this I would like to collaborate. I am pretty lost at where to start.I mostly don't understand the transition from single character classification (which I know how to do) to multi digit recognition.

For example it says that if you want to limit it to finding strings of up to 5 characters you should have 5 full connected classifiers. Does this mean that it should connect the last convolution layer to 5 classifiers that are each trained to recognize single digits? Wouldn't this result in them all just returning the same answer?

---

pkpp1233 **Jul 26**

@bri thanks. going to check out cs231n to dig into concepts. let me know if you want to start an email chain to work through project.

---

stephenwithav **Jul 27**

Thank you for that link.

---

raphael_189174082 **Jul 30**

I am interested in collaborating with anyone who is working using deep learning as the capstone project for the Udacity MLE program. How many would be interested in joining a google group?

---

pkpp1233 **Jul 30**

I'd be in! Share link to group plz.

---

e.berdibekov **Jul 30**

There was a long pause before I finished L4 assignments, not have a time after work, But last weekend started implementing this Capstone Project.
Want to share, as results:
1) Created generator, randomly collecting notMNIST characters to sequence in line, with applying rotation, etc other transformations, generated 64x64 images. Run it, and collect dataset of size 100K for train and 30K for test.

2) Trained on this dataset on latest model architecture: convolution layers (32, 64, 128, 256, 512), with pooling in 4/5 layers, everywhere kernels size is 5x5. Fully connected layer 2048-1024. Training took approximately 1.5 hours on my GPU (not even fast :)) Reached 92.7% on validation set, 92.1% test set.
It is my final result, adjusting hyperparams and increasing model capacity to reach highest val accuracy not is case for me.

3) After training on realistic SVHN dataset, realized that training was it is easier than notMNIST sequences
Reached on same model architecture 96.6% accuracy on validation and test sets after 32 epochs.

Finally collected 20 images of street numbers from internet. Each image represents different environment, light, perspective, materials and other conditions. Evaluation on this set shows 67.7% accuracy.

Actually preparation procedures was longest part in this project. Training can be for someone faster, if has powerful GPU.
If anyone interested, I can paste code to parse SVHN matlab 7.3 format file. It takes about 2 hours, mb can save your time

Update: Here my training approach was right, however I think used not appropriate accuracy function that uses "partial credit". For example: instead of 0% accuracy for prediction L=4 S=1,2,3,4 and labels L=4 S=1,2,8,4 my accuracy gives 80%. So task is not easy, no free lunch here Need to train with more sophisticated goal. With fixed accuracy function my model's validation set accuracy on realistic dataset is **82.2%**. Details in my posts below.

---

nishant_agrawal **Jul 31**
Forum Mentor

Have few questions on approach;
1. Did you use blank images for numbers having digits less than 3?
2. Does this model identify individual digits and then combines the output (and how) or does it identify whole sequence at one go like that in google paper?

I trained CNN on individual digits from notMNIST and got 97% with dropout, on AWS. Have you tried on g2.8 on AWS?

---

e.berdibekov **Jul 31**

Yeah, I've achieved 96.5% accuracy on separate notMNIST recognizing task, but on my local PC. I have an Google Compute instances for calculations, but not for udacity. Thanks for mentioning it 🙂

About questions:
1. Yes, in generator I took into account this case. Actually I've used uniform distribution to generated random lengths of sequences. Here more appropriate is to use truncated normal distribution (because in nature we see rare long sequences, often zero or 1-3 digits long street numbers), but can't implement that in python.
2. Just like in google paper. Feedforward network with outputs: seq length prediction, sequence of digits predictions.

That link little bit spoilers for us, do you think?
If I have not seen that paper, I'll be chose to use RNN. Latest our tasks were about RNNs.

---

nishant_agrawal **Jul 31**
Forum Mentor

From what I understand your implementation predicts the sequence in one go, instead of combining digits; is this correct?
How do you combine Length with sequence for training? How do you use length and sequence in target labels for training?

---

raphael_189174082 **Jul 31**

If anyone is interested in joining a Google group that supplements the forum, please send an email to tam.raphaelk@gmail.com. It is an invitation only group. I will invite anyone who pledges to abide by the Udacity Honor Code. We will organize study groups for those in the San Francisco Bay Area. We will have online discussions on materials that are not in Udacity's Deep Learning course. They include, but not limited to, **http://cs231n.github.io** and **http://www.deeplearningbook.org**. This group is for those who want to learn faster and complete the Capstone projects sooner by leveraging collaboration.

---

raphael_189174082 **Jul 31**

please send an email to tam.raphaelk@gmail.com.

---

bri **Aug 1**

@brian_20792649369148

Hi Brian ,

The approach I was thinking is that I would use three convolution layers for a basic convolution model. On top of the convolution model I'm adding a model for regression which would detect a digit within a 32x32 window. I am adding another model of top of the convolution base model that will use the sliding window approach to pick out the digits from the background. Still working on the implementation but I'll let you know how this approach works out.

---

e.berdibekov **Aug 3**

@nishant_agrawal ,

As I write in deleted post, because of maximizing log-likelihood you can add up your errors.
In google's paper they used probabilistic model:

$$P(\mathbf{S} = \mathbf{s}|X) = P(L = n \mid X)\Pi_{i=1}^{n} P(S_i = s_i \mid X).$$

In loss function they add error of L (length) to seqence errors. You can do this in one-go.

However, since you asked reasonable question, I do some research, checked correctness of my algorithm. Training procedure was right, but in accuracy evaluating function I used not appropriate accuracy function that uses "partial credit". For example: instead of 0% accuracy for prediction L=4 S=1,2,3,4 and labels L=4 S=1,2,8,4 my function gives 80% accuracy.
So, thank you for help :), I fixed my accuracy function:

```
def eval_accuracy(eval_l_preds, eval_preds, l_labels, labels, masks):
    concatted = np.concatenate((np.reshape((eval_l_preds == l_labels), [-1, 1]),
                                (eval_preds * masks) == labels), axis=1)
    return 100.0 * (np.sum([np.all(row) for row in concatted])) / len(labels)
```

Rechecked my results: 82.2% accuracy on realistic data validation set. Already working on increasing model capacity.

---

nishant_agrawal **Aug 4**
Forum Mentor

Hi **@e.berdibekov** ,

Thanks for confirming.
Google paper mentions - "Each of the softmax models (the model for L and each Si) can use exactly the same backprop learning rule as when training an isolated softmax layer, except that a digit classifier softmax model backprops nothing on examples for which that digit is not present."

How is this condition met in your model?
For example, if I have loss = tf.reduce_mean(softmax_len) + tf.reduce_mean(softmax_digit_1)+tf.reduce_mean(softmax_2)+ ....3....4...5
Do you mask reduce_mean terms 4 and 5, if those digits are not present?
What does your loss function look like?

---

e.berdibekov **Aug 5**

Hi **@nishant_agrawal** ,

My loss function is:

```
# training
pred_l_logits, pred_logits = model(tf_train_dataset, True)
loss = tf.nn.sparse_softmax_cross_entropy_with_logits(pred_l_logits, tf_train_lens)
for i in range(max_seq_len):
    loss += tf.nn.sparse_softmax_cross_entropy_with_logits(pred_logits[i], tf_train_la
loss = tf.reduce_mean(loss)
```

I'm masking my model's output before calculating errors. These last codes of model function where I use output masking:

```
    pred_l = tf.matmul(h2, outl_w) + outl_b
    ret = []
    for j in range(max_seq_len):
        if train:
            ret.append((tf.matmul(h2, outk_w[j]) + outk_b[j]) * tf_train_masks[j])
        else:
            ret.append(tf.matmul(h2, outk_w[j]) + outk_b[j])
```

---

nishant_agrawal **Aug 8**
Forum Mentor

Hi **@e.berdibekov** ,

Have you used 1-hot encoding for train and test labels?

---

e.berdibekov **26d**

@nishant_agrawal , on prepared batches I'm using index labels [0-9]. In training I'm calculating loss with `tf.nn.sparse_softmax_cross_entropy_with_logits`, so I don't need one-hot encoded samples to feed the model. In evaluation time immediately converting one-hot predictions to indexes using argmax.

STUDENT RESOURCES +

UDACITY +

INQUIRIES +