

< # announcement >



toyeiei 1/6/25, 11:44 PM

DSB11 Scholarships

@everyone สวัสดีครับทุกคน

นหนนหนนหนนหนนหนนหนน แอดมาแจกทุน
แล้วว เราเพิ่มเป็นทั้งหมด 250 ทุนเลยนะค้า
บบบบ แจกยับ ร้องขอชีวิตดตดตสำหรับ alumni
กรบบบบบบ

❤️ ตรวจสอบทุนได้ที่ (กด CTRL+F เลือชื่อ
Facebook Name ของตัวเองได้เลยนะค้าบ)
<https://datarockie.co/dsb11-lucky>

❤️ กรอกฟอร์มรับทุนได้ที่
<https://datarockie.co/dsb11-form>

ขอบคุณทุกคนมาากเลยนะครับที่ร่วมกิจกรรม
มมมม เพื่อนๆที่ได้ทุน มาลุยกันต่อรุ่น 11 นะค้า
บบบ

ปล. ข้าเลยเพราะ supermetrics มันตึง page
reviews ไม่ได้ แอดต้องเข้าไป scrape เอง ก่าๆ
555 จะร้องงงง

เพื่อนๆที่อ่าน โพสต์นี้แล้ว กด ✅ ให้แอดหน่อยนะ
ครับบบบบบ ขอคุณครับทุกคน เฮ้ (เพื่อน' ↓
ได้ทุน ไม่ต้องเสียใจนะค้าบ ปีนี้ออร์สฟรีเราเลย

This is a threads only channel.

Create Thread

DSB11 Scholarships

@everyone Helloooo everyoneeeeeeeeeee! I'm here to announce the scholarship results! We've increased the total number to 250 scholarships! Giving them away like crazy—help meeeeeee, alumni! Much respect 🙏🙏

❤️ Check the scholarship results here (Press CTRL+F and search for your Facebook name):
<https://datarockie.co/dsb11-lucky>

❤️ Fill out the form to claim your scholarship here:
<https://datarockie.co/dsb11-form>

Thank you so so much to everyone who joined the activity! 🥳
To those who got the scholarship—let's continue the journey in DSB11 together!

P.S. It was delayed because Supermetrics couldn't pull the page reviews, so I had to scrape them manually... what a pain 😩 I almost cried lol

If you've read this post, please hit ✅ so I know—thanks so much, everyone! Yay!

(And for those who didn't get a scholarship, don't be discouraged—we have tons of free courses this year, releasing every month. Stay tuned!)

```

def test(self, training_data, test_data, epochs, mini_batch_size,
        test_data=None):
    len(training_data)
    j in xrange(epochs):
        random.shuffle(training_data)
        mini_batches = [
            training_data[k:k+mini_batch_size]
            for k in xrange(0, len(training_data), mini_batch_size)]
        for mini_batch in mini_batches:
            if test_data:
                print "Epoch (%i): (%i) / (%i).for" % (j+1, len(mini_batches), len(training_data))
            else:
                j, self.evaluate(test_data),
                print "Epoch (%i) complete".format(j+1)
            la_b = [np.zeros(b.shape) for b in self.biases]
            la_w = [np.zeros(w.shape) for w in self.weights]
            x, y in mini_batch:
                delta_nabla_b, delta_nabla_w = self.backprop(x, y)
                nabla_b = [nb+delta for nb, delta in zip(nabla_b, delta_nabla_b)]
                nabla_w = [nw+delta for nw, delta in zip(nabla_w, delta_nabla_w)]
                f.weights = [w-(rate/len(mini_batch))*nw for w, nw in zip(self.weights, nabla_w)]
                f.biases = [b-(rate/len(mini_batch))*nb for b, nb in zip(self.biases, nabla_b)]
            kprop(self, x, y):
                la_b = [np.zeros(b.shape) for b in self.biases]
                la_w = [np.zeros(w.shape) for w in self.weights]
            def forward():
                activation = x
                activations = [x] # list to store all the activations on each layer
                ivations = [] # list to store all the ivations on each layer
                for b, w in zip(self.biases, self.weights):
                    z = np.dot(w, activation)+b
                    zs.append(z)
                    activation = sigmoid(z)
                    activations.append(activation)
                activations.append(activation)
            def backward_pass():
                delta = self.cost_derivative(activations[-1])
                sigmoid_prime(zs[-1])
                la_b[-1] = delta
                la_w[-1] = np.dot(delta, activations[-2])
                l in xrange(2, self.num_layers):
                    z = zs[-l]
                    sp = sigmoid_prime(z)
                    delta = np.dot(self.weights[l+1: self.num_layers],
                                activations[l+1])
                    la_b[-l] = np.dot(delta, activations[l+1])
                    la_w[-l] = np.dot(delta, activations[l+2])
            def feedforward(self, test_data):
                results = []
                for x in test_data:
                    results.append(self.feedforward(x))
                return results

```

Tharaphorn Kanchanasri	0.00242000066
H	2
K	3
C	7
A	2
S	9
F	9
E	1
N	3
N	2
F	3
F	1
C	3
N	5
K	3
N	9
V	1
J	9
K	3
S	9
V	3
S	4
F	5
N	7
J	3
F	9
C	7

❤️ Check the scholarship results here
 (Press CTRL+F and search for your
 Facebook name):
<https://datarockie.co/dsb11-lucky>

Naphon Santisukwongchot

0.1062857378

