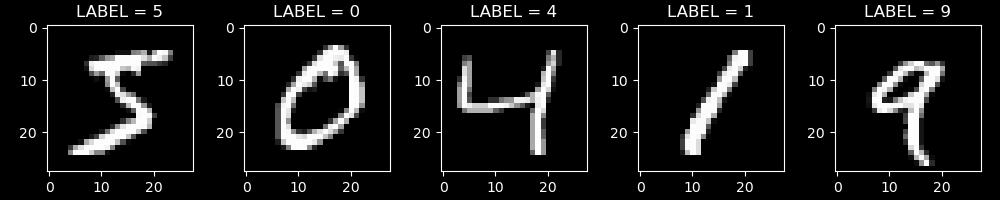
# گام اول – دریافت دیتاست

برای دریافت مجموعه داده، تابع get\_data را تعریف میکنیم که ساختار کلی آن از لینک موجود در دستور کار وجود دارد و تغییراتی جزئی روی آن صورت گرفته است. در این تابع پس از استخراج داده ها به عنوان نمونه، ۵ داده اول را پلات میکنیم و لیبل آن را در پلات نمایش میدهیم:



# گام دوم – محاسبه‌ی خروجی

برای پیاده‌سازی شبکه عصبی خود، کلاسی تحت عنوان NeuralNetwork ایجاد میکنیم و اطلاعاتی نظیر وزن ها، بایاس ها و سایر موارد مورد نیاز را به عنوان فیلد های آن تعریف میکنیم. یکی از متد های این کلاس feedforward است که به صورت زیر میباشد:

    def feedforward(self, img):

        z1 = (weights[0] @ img[0]) + biases[0]

        a1 = np.asarray([sigmoid(z[0]) for z in z1]).reshape((16, 1))

        z2 = (weights[1] @ a1) + biases[1]

        a2 = np.asarray([sigmoid(z[0]) for z in z2]).reshape((16, 1))

        z3 = (weights[2] @ a2) + biases[2]

        a3 = np.asarray([sigmoid(z[0]) for z in z3]).reshape((10, 1))

        return [a1, a2, a3], [z1, z2, z3]

که a3 خروجی شبکه میباشد. حال به کمک تابع calculate\_accuracy دقت را محاسبه میکنیم:

def calculate\_accuracy(self):

        number\_of\_correct\_guesses = 0

        for image in range(self.number\_of\_samples):

            guess = np.argmax(self.feedforward(self.train\_set[image])[0][-1])

            label = np.argmax(self.train\_set[image][1])

            number\_of\_correct\_guesses = number\_of\_correct\_guesses + 1 if guess == label else number\_of\_correct\_guesses

        return number\_of\_correct\_guesses / self.number\_of\_samples

برای ۱۰۰ داده اول، دقت نزدیک عدد ۱۰ درصد میباشد. در یک نمونه اجرا خروجی به صورت زیر بدست آمد:

        STEP 2: CALCULATING INITIAL ACCURACY

            initial accuracy: 15.0%

# گام ۳ – پیاده‌سازی backpropagation

به کمک متد هایی که داخل کلاس NeuralNetwork تعریف شدند (مانند feedforward و back\_propagation) ، شبه کد موجود در دستور کار را داخل train\_network پیاده سازی کردیم. در هر لحظه شماره epoch، شماره batch و همچنین شماره عکسی که در حال پردازش آن هستیم در خروجی نمایش داده میشود و در پایان زمان، دقت، و همچنین نمودار میزان خطا برای epoch ها مختلف نمایش داده میشود.

TRAINING THE NETWORK:

    EPOCH: 01/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 02/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 03/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 04/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 05/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 06/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 07/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 08/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 09/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 10/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 11/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 12/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 13/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 14/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 15/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 16/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 17/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 18/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 19/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    EPOCH: 20/20     BATCH: 010/010   IMAGE: 0010/0010    EPOCH COMPLETED!

    TRAINING PROCESS COMPLETED IN 201S

    THE ACCURACY OF THE NETWORK IS 36.0%



# گام ۴ – vectorization