Bahri Batuhan Bilecen
22201372
4.12.2023

EEE543 Mini Project 2 – RNN on Human Activity Recognition

1. Problem Definition

In this work, we are required to train a recurrent neural network (RNN) on a human activity recognition (HAR) dataset, with the backpropagation through time (BPTT) algorithm. Some ablation studies regarding the hyperparameters of the network and the training regime are also demonstrated with accompanying conclusions in the following sections.

2. Derivations and Implementation Details

The derivations for backpropagation through time (BPTT) for recurrent neural networks and the usage of truncated BPTT as batch learning are given in this section.

2.1. BPTT

This section includes a comprehensive derivation for BPTT using matrix notations.

$\bar{\mathbf{M}}_{\mathrm{K}\times(\mathrm{M}+1)}$ is the extended matrix where the +1 dimension is padded with extra 1's to accommodate for biases, and $\mathbf{M}_{\mathrm{K}\times\mathrm{M}}$ is its non-extended version.

The RNN equations are as follows:

$$\mathbf{h}^{\mathrm{t}}_{50\times1} = \tanh\left(\bar{\mathbf{W}}_{\mathrm{ih}_{50\times(3+1)}}\bar{\mathbf{i}}^{\mathrm{t}}_{(3+1)\times1} + \bar{\mathbf{W}}_{\mathrm{hh}_{50\times(50+1)}}\bar{\mathbf{h}}^{\mathrm{t}-1}_{(50+1)\times1}\right)$$

$$\mathbf{y}_{6\times1} = \mathrm{sigmoid}(\bar{\mathbf{W}}_{\mathrm{ho}_{6\times(50+1)}}\bar{\mathbf{h}}^{\mathrm{t}}_{(50+1)\times1})$$

The error is determined as the multicategory cross-entropy loss:

$$\mathrm{E} = -\sum_{\mathrm{k}\in\mathrm{classes}} \mathrm{d_k}\log(\mathrm{y_k}) + (1-\mathrm{d_k})\log(1-\mathrm{y_k}) \equiv -\log(\mathrm{y_k})$$

Finding the derivative with respect to $\mathbf{W}_{\mathrm{ho}}$ is straightforward:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\mathrm{ho}}} = \frac{\partial \mathbf{E}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{o}}\frac{\partial \mathbf{o}}{\partial \mathbf{W}_{\mathbf{ho}}} = -\frac{1}{\mathbf{y}_{6\times1}}\left(\mathbf{y}_{6\times1}\otimes(\mathbf{d}_{6\times1}-\mathbf{y}_{6\times1})\right)\left(\bar{\mathbf{h}}^{\mathrm{last}}_{(50+1)\times1}\right)^{\mathrm{T}} = (\mathbf{y}_{6\times1}-\mathbf{d}_{6\times1})\left(\bar{\mathbf{h}}^{\mathrm{last}}_{(50+1)\times1}\right)^{\mathrm{T}}$$

Note that in $\frac{\partial \mathbf{y}^{\mathrm{t}}}{\partial \mathbf{o}^{\mathrm{t}}} = \left(\mathbf{y}_{6\times1}\otimes(\mathbf{d}_{6\times1}-\mathbf{y}_{6\times1})\right)$, $\otimes$ is the Kronecker product. We either get $(\mathbf{y}_{6\times1})$ or $(\mathbf{1}-\mathbf{y}_{6\times1})$ from $\mathbf{y}_{6\times1}-\mathbf{d}_{6\times1}$.

For $\mathbf{W}_{\mathrm{hh}}$, we will use backpropagation through time (BPTT):

$$\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\mathrm{hh}}} = \frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\mathrm{last}}}\left(\sum_{\mathrm{t}}\frac{\partial \mathbf{h}^{\mathrm{last}}}{\partial \mathbf{h}^{\mathrm{last}-\mathrm{t}}}\frac{\partial \mathbf{h}^{\mathrm{last}-\mathrm{t}}}{\partial \mathbf{W}_{\mathrm{hh}}}\right)$$
$$= \frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\mathrm{last}}}\frac{\partial \mathbf{h}^{\mathrm{last}}}{\partial \mathbf{W}_{\mathrm{hh}}} + \frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\mathrm{last}}}\frac{\partial \mathbf{h}^{\mathrm{last}}}{\partial \mathbf{h}^{\mathrm{last}-1}}\frac{\partial \mathbf{h}^{\mathrm{last}-1}}{\partial \mathbf{W}_{\mathrm{hh}}} + \frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\mathrm{last}}}\frac{\partial \mathbf{h}^{\mathrm{last}}}{\partial \mathbf{h}^{\mathrm{last}-1}}\frac{\partial \mathbf{h}^{\mathrm{last}-1}}{\partial \mathbf{h}^{\mathrm{last}-2}}\frac{\partial \mathbf{h}^{\mathrm{last}-2}}{\partial \mathbf{W}_{\mathrm{hh}}} + \cdots$$

To understand better, write down several terms and observe the patterns:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\mathrm{last}}}\frac{\partial \mathbf{h}^{\mathrm{last}}}{\partial \mathbf{W}_{\mathrm{hh}}}$$

$$= \left(\text{diag}\left(\mathbf{1} - \mathbf{h}_{50\times1}^{\text{last}\ 2}\right)\left(\mathbf{W}_{\text{ho}_{6\times50}}\right)^{\text{T}}(\mathbf{y}_{6\times1} - \mathbf{d}_{6\times1})\left(\bar{\mathbf{h}}_{(50+1)\times1}^{\text{last}-1}\right)^{\text{T}}\right)_{50\times(50+1)}$$

$$= \mathbf{A}_{50\times1}\left(\bar{\mathbf{h}}_{(50+1)\times1}^{\text{last}-1}\right)^{\text{T}}$$

where $\text{diag}\left(\mathbf{1} - \mathbf{h}_{50\times1}^{\text{last}\ 2}\right)$ is the derivative of tanh with the output $\mathbf{h}_{50\times1}^{\text{last}}$.

Expanding the second and the third term yields:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\text{last}}}\frac{\partial \mathbf{h}^{\text{last}}}{\partial \mathbf{h}^{\text{last}-1}}\frac{\partial \mathbf{h}^{\text{last}-1}}{\partial \mathbf{W}_{\text{hh}}}$$

$$= \left(\text{diag}\left(\mathbf{1} - \mathbf{h}_{50\times1}^{\text{last}-1\ 2}\right)\mathbf{W}_{\text{hh}_{50\times50}}\text{diag}\left(\mathbf{I} - \mathbf{h}_{50\times1}^{\text{last}\ 2}\right)\left(\mathbf{W}_{\text{ho}_{6\times50}}\right)^{\text{T}}(\mathbf{y}_{6\times1} - \mathbf{d}_{6\times1})\left(\bar{\mathbf{h}}_{(50+1)\times1}^{\text{last}-2}\right)^{\text{T}}\right)_{50\times(50+1)}$$

$$= \text{diag}\left(\mathbf{1} - \mathbf{h}_{50\times1}^{\text{last}-1\ 2}\right)\mathbf{W}_{\text{hh}_{50\times50}}\mathbf{A}_{50\times1}\left(\bar{\mathbf{h}}_{(50+1)\times1}^{\text{last}-2}\right)^{\text{T}} = \mathbf{B}_{50\times1}\left(\bar{\mathbf{h}}_{(50+1)\times1}^{\text{last}-2}\right)^{\text{T}}$$

$$\frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\text{last}}}\frac{\partial \mathbf{h}^{\text{last}}}{\partial \mathbf{h}^{\text{last}-1}}\frac{\partial \mathbf{h}^{\text{last}-1}}{\partial \mathbf{h}^{\text{last}-2}}\frac{\partial \mathbf{h}^{\text{last}-2}}{\partial \mathbf{W}_{\text{hh}}}$$

$$= \text{diag}\left(\mathbf{1} - \mathbf{h}_{50\times1}^{\text{last}-2\ 2}\right)\mathbf{W}_{\text{hh}_{50\times50}}\mathbf{B}_{50\times1}\left(\bar{\mathbf{h}}_{(50+1)\times1}^{\text{last}-3}\right)^{\text{T}} = \mathbf{C}_{50\times1}\left(\bar{\mathbf{h}}_{(50+1)\times1}^{\text{last}-3}\right)^{\text{T}}$$

While propagating through time, we will utilize previously found matrices ($\mathbf{A}_{50\times1}, \mathbf{B}_{50\times1}, \mathbf{C}_{50\times1}$ ...). This will become handy while coding.

For $\mathbf{W}_{\text{ih}}$, we will again use BPTT:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ih}}} = \frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\text{last}}}\left(\sum_{\text{t}}\frac{\partial \mathbf{h}^{\text{last}}}{\partial \mathbf{h}^{\text{last}-\text{t}}}\frac{\partial \mathbf{h}^{\text{last}-\text{t}}}{\partial \mathbf{W}_{\text{ih}}}\right)$$

$$= \frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\text{last}}}\frac{\partial \mathbf{h}^{\text{last}}}{\partial \mathbf{W}_{\text{ih}}} + \frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\text{last}}}\frac{\partial \mathbf{h}^{\text{last}}}{\partial \mathbf{h}^{\text{last}-1}}\frac{\partial \mathbf{h}^{\text{last}-1}}{\partial \mathbf{W}_{\text{ih}}} + \frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\text{last}}}\frac{\partial \mathbf{h}^{\text{last}}}{\partial \mathbf{h}^{\text{last}-1}}\frac{\partial \mathbf{h}^{\text{last}-1}}{\partial \mathbf{h}^{\text{last}-2}}\frac{\partial \mathbf{h}^{\text{last}-2}}{\partial \mathbf{W}_{\text{ih}}} + \cdots$$

$$\frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\text{last}}}\frac{\partial \mathbf{h}^{\text{last}}}{\partial \mathbf{W}_{\text{ih}}}$$

$$= \left(\text{diag}\left(\mathbf{1} - \mathbf{h}_{50\times1}^{\text{last}\ 2}\right)\left(\mathbf{W}_{\text{ho}_{6\times50}}\right)^{\text{T}}(\mathbf{y}_{6\times1} - \mathbf{d}_{6\times1})\left(\bar{\mathbf{i}}_{(3+1)\times1}^{\text{last}-1}\right)^{\text{T}}\right)_{50\times(3+1)} = \mathbf{A}_{50\times1}\left(\bar{\mathbf{i}}_{(3+1)\times1}^{\text{last}-1}\right)^{\text{T}}$$

$$\frac{\partial \mathbf{E}}{\partial \mathbf{h}^{\text{last}}}\frac{\partial \mathbf{h}^{\text{last}}}{\partial \mathbf{h}^{\text{last}-1}}\frac{\partial \mathbf{h}^{\text{last}-1}}{\partial \mathbf{W}_{\text{ih}}}$$

$$= \left(\text{diag}\left(\mathbf{1} - \mathbf{h}_{50\times1}^{\text{last}-1\ 2}\right)\mathbf{W}_{\text{hh}_{50\times50}}\text{diag}\left(\mathbf{I} - \mathbf{h}_{50\times1}^{\text{last}\ 2}\right)\left(\mathbf{W}_{\text{ho}_{6\times50}}\right)^{\text{T}}(\mathbf{y}_{6\times1} - \mathbf{d}_{6\times1})\left(\bar{\mathbf{i}}_{(3+1)\times1}^{\text{last}-2}\right)^{\text{T}}\right)_{50\times(3+1)}$$

$$= \text{diag}\left(\mathbf{1} - \mathbf{h}_{50\times1}^{\text{last}-1^2}\right)\mathbf{W}_{\text{hh}_{50\times50}}\mathbf{A}_{50\times1}\left(\bar{\mathbf{i}}_{(3+1)\times1}^{\text{last}-2}\right)^{\text{T}} = \mathbf{B}_{50\times1}\left(\bar{\mathbf{i}}_{(3+1)\times1}^{\text{last}-2}\right)^{\text{T}}$$

Several iterations reveal that the only difference with $\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{hh}}}$ is that the rightest matrix is $\bar{\mathbf{i}}_{(3+1)\times1}^{\text{k}}$ rather than $\bar{\mathbf{h}}_{(50+1)\times1}^{\text{k}}$.

2.2. Truncated BPTT

In RNNs, batch learning is usually done with truncated BPTT. The difference between BPTT and truncated BPTT is that the forward and backward passes are "truncated", so that the weight updates are done more frequently, avoiding vanishing/exploding gradients and encouraging short-term memory.

The important thing in truncated BPTT is that the last hidden states are saved at the end of each batch, later to be used in the next batch as the initial condition (see line 5 of **Algorithm 1**).

(a) BPTT (b) Truncated BPTT

Figure 1. BPTT vs Truncated BPTT. Right and left arrows denote forward and backward passes, respectively. Figure adapted from the lecture notes.

2.3. Implementation Details

The implementation details are briefly given below:

- The implementation is made from scratch; hence the matrix notations are compliant with the course content. I utilized **`torch`** library to migrate the data to GPU and perform the training sessions faster with **`CUDA`**, but I did not utilize any of its automatic gradient functionalities. In other words, using **`torch`** is identical to using **`numpy`** but with GPU in this scenario; and in fact, most of the matrix functions are the same in **`torch`** & **`numpy`** in terms of valid input arguments and operation behavior.
- For stability purposes and to avoid NaNs, I **clipped the weight gradients between [-1,1].**

The overall procedure using the derivations given in **Section 2.1** is given in **Algorithm 1.**

Algorithm 1. Pseudocode for training an RNN with truncated BPTT

Initialize all extended weights $\bar{\mathbf{W}}$ with uniform distribution between [-0.01,0.01]
for epoch in all epochs do
  for sample $(\mathbf{i}, \mathbf{d})$ in training dataset do # data and labels
    Divide $\mathbf{i}$ into batches for truncated BPTT
    Initialize $\bar{\mathbf{h}}^{\text{prev}}_{(50+1)\times 1}$ with 0 if first batch, else $\bar{\mathbf{h}}^{\text{prev}}_{(50+1)\times 1} \leftarrow \bar{\mathbf{h}}^{\text{seq\_length}}_{(50+1)\times 1}$
    for t=0:seq_length do # **Forward pass**
      $\mathbf{h}^{\text{curr}}_{50\times 1} \leftarrow \tanh(\bar{\mathbf{W}}_{\text{ih}_{50\times(3+1)}} \bar{\mathbf{i}}^{\text{t}}_{(3+1)\times 1} + \bar{\mathbf{W}}_{\text{hh}_{50\times(50+1)}} \bar{\mathbf{h}}^{\text{prev}}_{(50+1)\times 1})$
      $\bar{\mathbf{h}}^{\text{prev}}_{(50+1)\times 1} \leftarrow \bar{\mathbf{h}}^{\text{curr}}_{(50+1)\times 1}$
      Save $\bar{\mathbf{h}}^{\text{curr}}_{(50+1)\times 1}$ at t and $\bar{\mathbf{i}}^{\text{t}}_{(3+1)\times 1}$ for backprop
    end for
    $\mathbf{y}_{6\times 1} \leftarrow \text{sigmoid}(\bar{\mathbf{W}}_{\text{ho}_{6\times(50+1)}} \bar{\mathbf{h}}^{\text{curr}}_{(50+1)\times 1})$
    $\mathbf{E} = -\frac{1}{\text{seq\_length}} \sum_{\text{seq\_length}} \log(\boldsymbol{y})$ # log-likelihood error
    $\frac{\partial \mathbf{E}}{\partial \mathbf{h}} \leftarrow \left(\mathbf{W}_{\text{ho}_{6\times 50}}\right)^{\text{T}} (\mathbf{y}_{6\times 1} - \mathbf{d}_{6\times 1})$
    $\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ho}}} \leftarrow \frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ho}}} + (\mathbf{y}_{6\times 1} - \mathbf{d}_{6\times 1}) \left(\bar{\mathbf{h}}^{\text{seq\_length}}_{(50+1)\times 1}\right)^{\text{T}}$ **# BPTT for Who**
    for t=seq_length:0 do # **Backward pass**
      $\frac{\partial \mathbf{E}}{\partial \mathbf{h}} \leftarrow \text{diag}\left(\mathbf{1} - \mathbf{h}^{\text{t}}_{50\times 1}{}^{2}\right) \frac{\partial \mathbf{E}}{\partial \mathbf{h}}$ **# Formation of the intermediate matrices A,B,C…**
      $\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{hh}}} \leftarrow \frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{hh}}} + \frac{\partial \mathbf{E}}{\partial \mathbf{h}} \left(\bar{\mathbf{h}}^{\text{t-1}}_{(50+1)\times 1}\right)^{\text{T}}$ **# BPTT for Whh**
      $\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ih}}} \leftarrow \frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ih}}} + \frac{\partial \mathbf{E}}{\partial \mathbf{h}} \left(\bar{\mathbf{i}}^{\text{t-1}}_{(3+1)\times 1}\right)^{\text{T}}$ **# BPTT for Wih**
      $\frac{\partial \mathbf{E}}{\partial \mathbf{h}} \leftarrow \mathbf{W}_{\text{hh}_{50\times 50}} \frac{\partial \mathbf{E}}{\partial \mathbf{h}}$ **# Formation of the intermediate matrices A,B,C…**
      Save $\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ho}}}$, $\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{hh}}}$, $\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ih}}}$ for batch update
    Clip gradients to avoid explosion
    end for
    $\bar{\mathbf{W}}_{\text{ih}_{(50)\times(3+1)}} \leftarrow \bar{\mathbf{W}}_{\text{ih}_{(50)\times(3+1)}} - \eta \left(\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ih}}}\right)$ **# Gradient descent**
    $\bar{\mathbf{W}}_{\text{ho}_{6\times(50+1)}} \leftarrow \bar{\mathbf{W}}_{\text{ho}_{6\times(50+1)}} - \eta \left(\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{ho}}}\right)$
    $\bar{\mathbf{W}}_{\text{hh}_{(50)\times(50+1)}} \leftarrow \bar{\mathbf{W}}_{\text{hh}_{(50)\times(50+1)}} - \eta \left(\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{\text{hh}}}\right)$
    end if
    Track $\mathbf{E}$ until convergence
  end for
  shuffle training dataset
end for

Test and validation steps are not given in the pseudocode. For more details, please check the attached .py file.

3. Results and Discussion

This section includes all results and conclusions.

3.1. Effects of learning rate, hidden layer size, and mini batch size without validation set (b)

The experimental results are given in Table 1 and Figure 2-4. LR, N, and B denote learning rate, hidden layer size, and mini batch size, respectively.

Table 1. Top-k scores for the models trained in Part (b). The best model is given in bold.

LR	N	B	Top-1	Top-2	Top-3	Top-4	Top-5
0.1	50	10	0.410	0.603	0.762	0.928	0.977
0.1	**50**	**30**	**0.582**	**0.727**	**0.860**	0.962	0.990
0.1	100	10	0.342	0.568	0.717	0.865	0.956
0.1	100	30	0.353	0.540	0.708	0.838	0.997
0.05	50	10	0.529	0.717	0.857	0.960	0.977
0.05	50	30	0.405	0.585	0.713	0.896	0.983
0.05	100	10	0.353	0.547	0.718	0.940	0.970
0.05	100	30	0.318	0.493	0.642	0.753	0.813

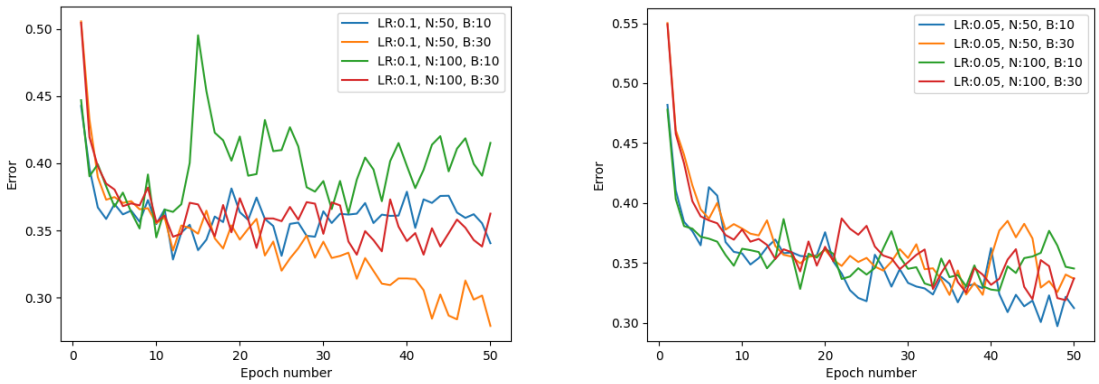


Figure 2. Comparing the models on learning rates (0.1 on left, 0.05 on right).

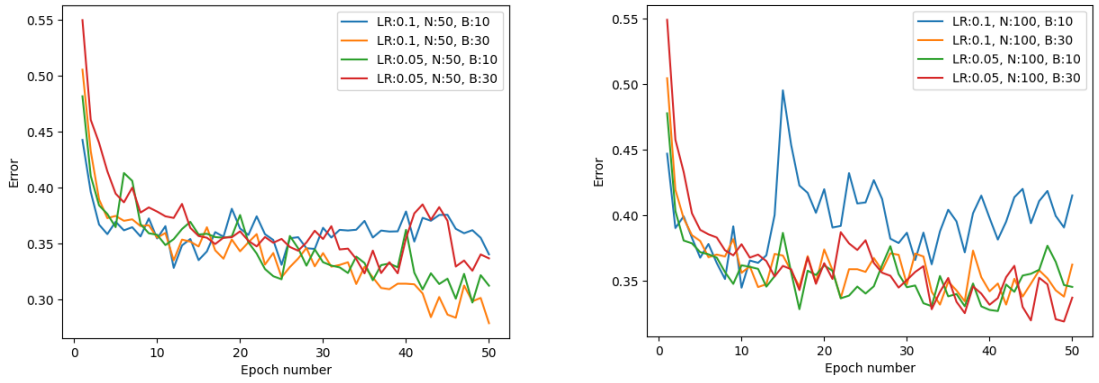


Figure 3. Comparing the models on hidden layer sizes (50 on left, 100 on right). Best viewed digitally.

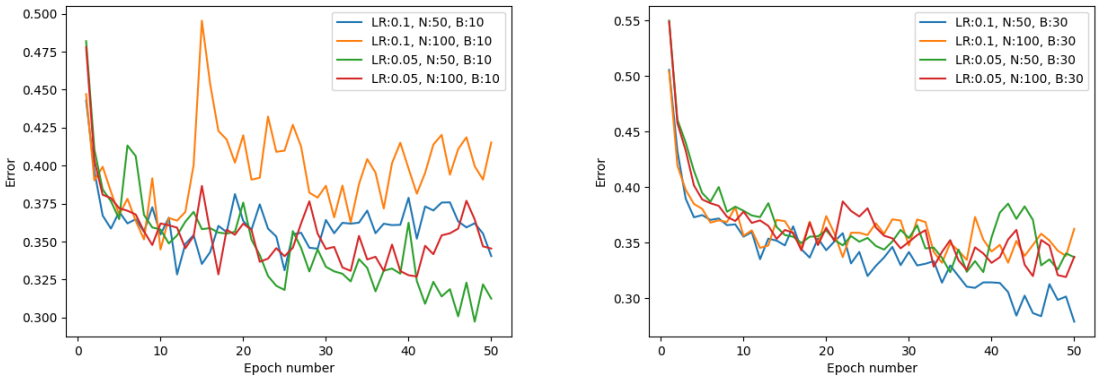
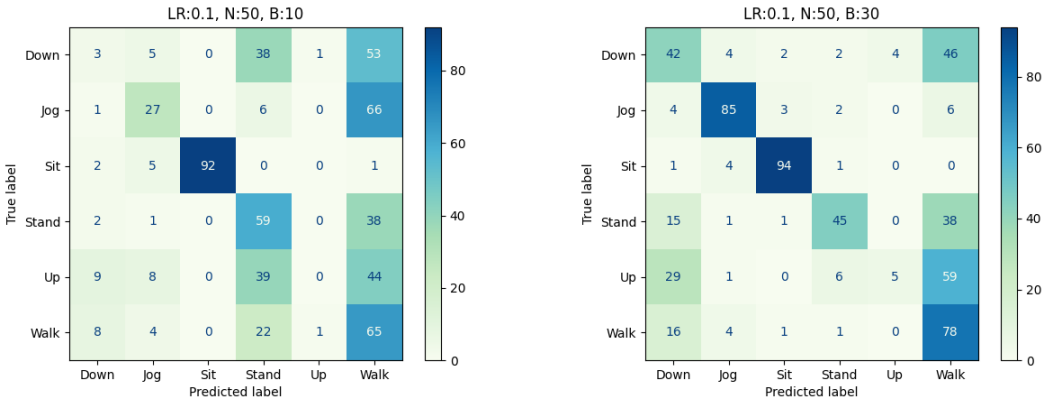


Figure 4. Comparing the models on mini batch sizes (10 on left, 30 on right).

- **Figure 2** reveals that higher batch size with a smaller number of hidden layers work the best among the tried configurations for learning rate = 0.1 (left plot). Higher learning rate enables the models converge to a point with less training error but is more unstable compared to the ones with lower learning rate.
- **Figure 3** suggests using a lower learning rate when model size increases (notice the blue curve on the right plot).
- **Figure 4** reveals using a larger batch size (hence enabling longer-term memory) may be more stable during training (right plot), especially if the learning rate is larger.



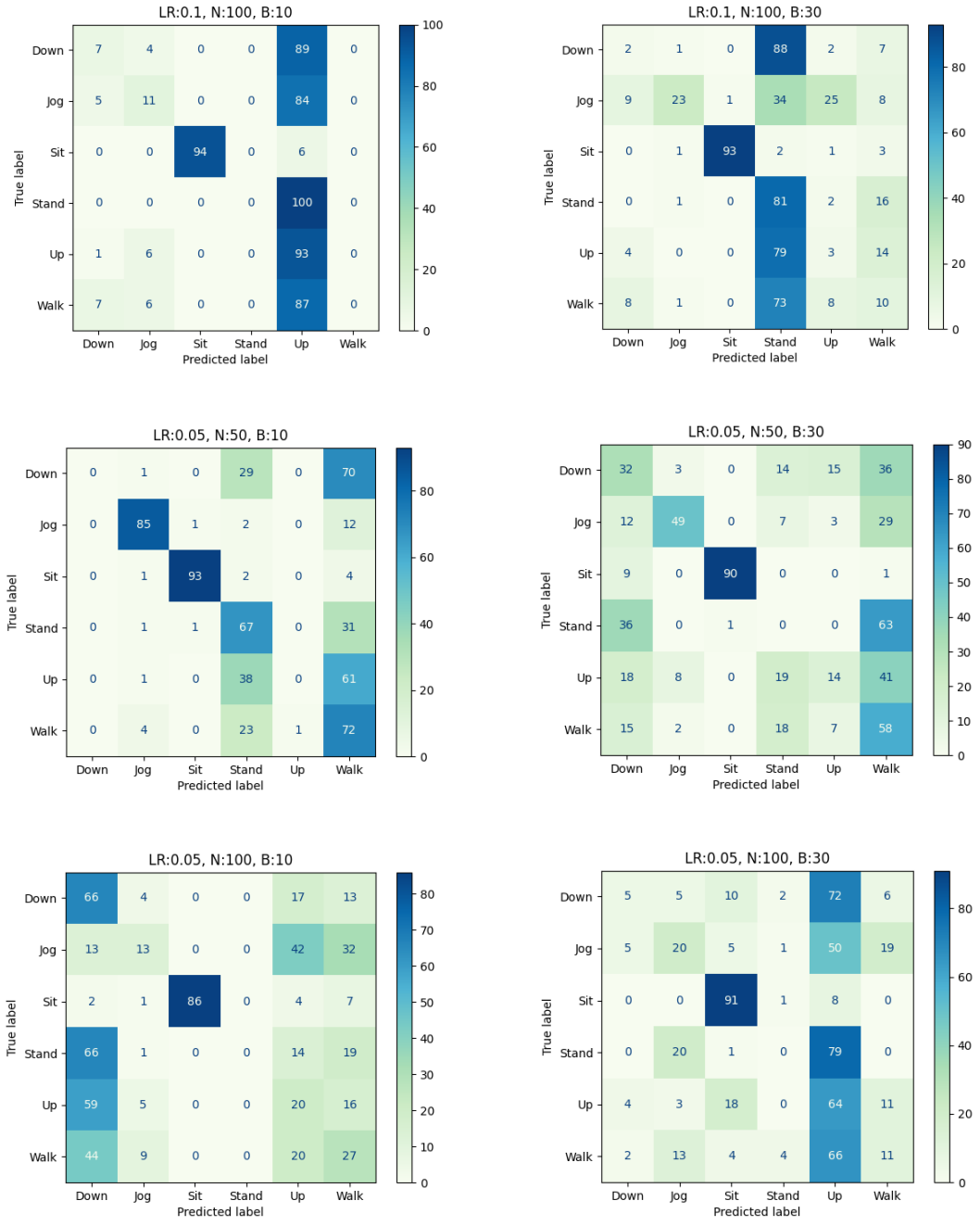


Figure 5. Confusion matrices for all trained 8 models in Part (b). The ideal case is to get a diagonal matrix.

- All models got the sitting motion correct. Notice some models are biased towards a specific motion due to bad training. The best model (LR:0.1, N:50, B:30) is the closest to a diagonal confusion matrix.
- Walking is sometimes mixed up with all other actions expect sitting - which makes sense.
- The distribution of the dataset classes (i.e., inherent bias) may play a role in the apparent biases in the confusion matrices as well.

3.2. Early stopping with validation set (c)

Table 2. Top-k scores for the models trained in Part (c). The improvements with early stopping compared to without early stopping are shown in bold, and worse results are shown in italic. Note that the Top-6 score is always 1.

			Without early stopping (last checkpoint)					With early stopping					
LR	**N**	**B**	**Top-1**	**Top-2**	**Top-3**	**Top-4**	**Top-5**	**Top-1**	**Top-2**	**Top-3**	**Top-4**	**Top-5**	**Early Stop Epoch**
0.1	50	10	0.358	0.525	0.706	0.872	0.975	**0.483**	**0.657**	**0.843**	**0.965**	**0.987**	38
0.1	50	30	0.555	0.732	0.867	0.980	0.995	**0.555**	**0.732**	**0.867**	**0.980**	**0.995**	50
0.1	100	10	0.418	0.613	0.730	0.855	0.938	**0.418**	**0.625**	**0.770**	**0.915**	**0.972**	12
0.1	100	30	0.450	0.652	0.785	0.912	0.977	**0.480**	**0.677**	**0.810**	**0.945**	*0.973*	35
0.05	50	10	0.410	0.675	0.792	0.888	0.915	**0.410**	**0.675**	**0.792**	**0.888**	**0.915**	50
0.05	50	30	0.522	0.645	0.783	0.955	0.973	**0.522**	**0.645**	**0.783**	**0.955**	**0.973**	50
0.05	100	10	0.490	0.626	0.813	0.913	0.925	*0.440*	*0.622*	*0.780*	**0.923**	**0.978**	46
0.05	100	30	0.283	0.480	0.667	0.883	0.980	**0.283**	**0.480**	**0.667**	**0.883**	**0.980**	50

Table 3. Top-k scores for the best models in Part (b) and Part (c). Note that the Top-6 score is always 1.

			Best results of Part (b)					Best results of Part (c)				
LR	**N**	**B**	**Top-1**	**Top-2**	**Top-3**	**Top-4**	**Top-5**	**Top-1**	**Top-2**	**Top-3**	**Top-4**	**Top-5**
0.1	50	10	0.410	0.603	0.762	0.928	0.977	**0.483**	**0.657**	**0.843**	**0.965**	**0.987**
0.1	50	30	**0.582**	0.727	0.860	0.962	0.990	0.555	**0.732**	**0.867**	**0.980**	**0.995**
0.1	100	10	0.342	0.568	0.717	0.865	0.956	**0.418**	**0.625**	**0.770**	**0.915**	**0.972**
0.1	100	30	0.353	0.540	0.708	0.838	0.997	**0.480**	**0.677**	**0.810**	**0.945**	0.973
0.05	50	10	**0.529**	**0.717**	**0.857**	**0.960**	**0.977**	0.410	0.675	0.792	0.888	0.915
0.05	50	30	0.405	0.585	0.713	0.896	0.983	**0.522**	**0.645**	**0.783**	**0.955**	0.973
0.05	100	10	0.353	0.547	0.718	**0.940**	0.970	**0.440**	**0.622**	**0.780**	0.923	**0.978**
0.05	100	30	**0.318**	**0.493**	0.642	0.753	0.813	0.283	0.480	**0.667**	**0.883**	**0.980**

- **Table 2** reveals the positive effect of early stopping. Since training RNNs are not very stable, in most of the considered hyperparameter combinations the early stopping helps.
- **Table 3** combines the best results of Part (b) and (c). The improvements are thanks to early stopping, and worse results may be due to the decrease in the training samples.

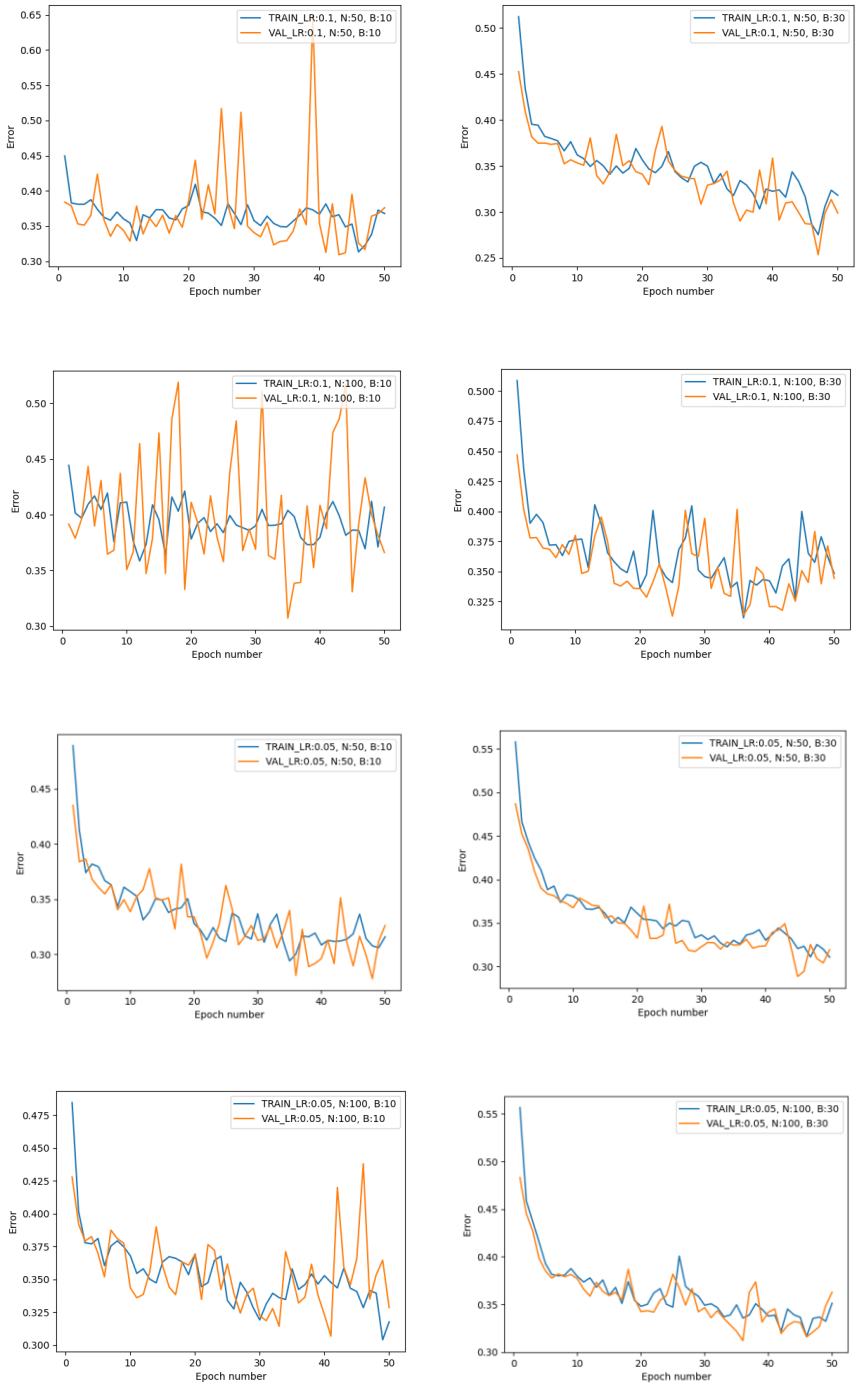
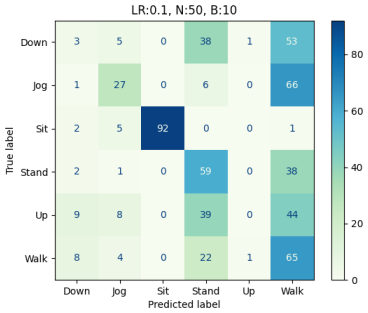
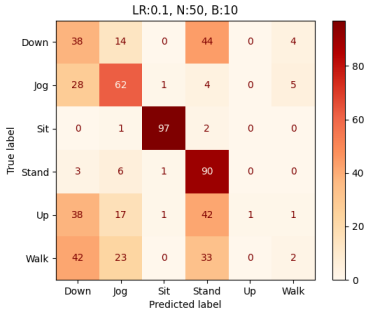
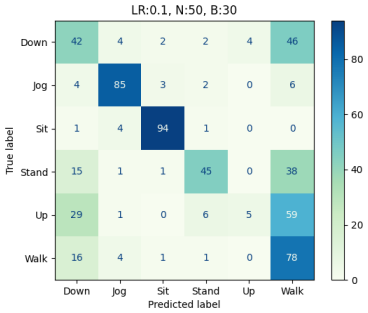


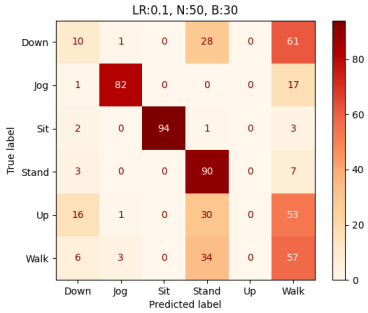
Figure 6. Training loss versus validation loss for all trained models in Part (c).

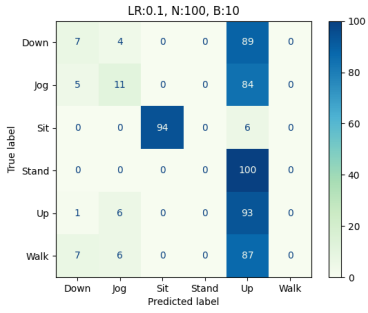
- Notice the validation versus training loss curve may suggest about the stability of the training besides the overfitting problem (LR:0.1, N:50, B:10 & LR:0.1, N:100, B:10).
- Some models could have been trained for more than 50 epochs since the validation curve does not show any sign of overfitting (LR:0.05, N:50, B:30 & LR:0.05, N:100, B:30).

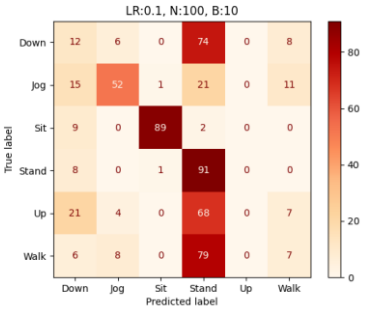

LR:0.1, N:50, B:10
Down 3 5 0 38 1 53
Jog 1 27 0 6 0 66
Sit 2 5 92 0 0 1
Stand 2 1 0 59 0 38
Up 9 8 0 39 0 44
Walk 8 4 0 22 1 65
True label
Down Jog Sit Stand Up Walk
Predicted label
0 20 40 60 80

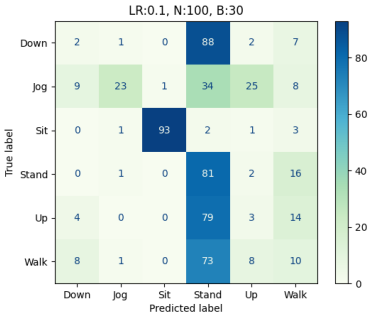

LR:0.1, N:50, B:10
Down 38 14 0 44 0 4
Jog 28 62 1 4 0 5
Sit 0 1 97 2 0 0
Stand 3 6 1 90 0 0
Up 38 17 1 42 1 1
Walk 42 23 0 33 0 2
True label
Down Jog Sit Stand Up Walk
Predicted label
0 20 40 60 80

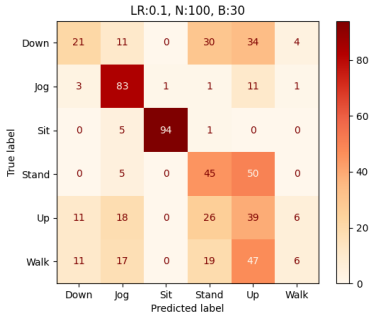

LR:0.1, N:50, B:30
Down 42 4 2 2 4 46
Jog 4 85 3 2 0 6
Sit 1 4 94 1 0 0
Stand 15 1 1 45 0 38
Up 29 1 0 6 5 59
Walk 16 4 1 1 0 78
True label
Down Jog Sit Stand Up Walk
Predicted label
0 20 40 60 80


LR:0.1, N:50, B:30
Down 10 1 0 28 0 61
Jog 1 82 0 0 0 17
Sit 2 0 94 1 0 3
Stand 3 0 0 90 0 7
Up 16 1 0 30 0 53
Walk 6 3 0 34 0 57
True label
Down Jog Sit Stand Up Walk
Predicted label
0 20 40 60 80


LR:0.1, N:100, B:10
Down 7 4 0 0 89 0
Jog 5 11 0 0 84 0
Sit 0 0 94 0 6 0
Stand 0 0 0 0 100 0
Up 1 6 0 0 93 0
Walk 7 6 0 0 87 0
True label
Down Jog Sit Stand Up Walk
Predicted label
0 20 40 60 80 100


LR:0.1, N:100, B:10
Down 12 6 0 74 0 8
Jog 15 52 1 21 0 11
Sit 9 0 89 2 0 0
Stand 8 0 1 91 0 0
Up 21 4 0 68 0 7
Walk 6 8 0 79 0 7
True label
Down Jog Sit Stand Up Walk
Predicted label
0 20 40 60 80


LR:0.1, N:100, B:30
Down 2 1 0 88 2 7
Jog 9 23 1 34 25 8
Sit 0 1 93 2 1 3
Stand 0 1 0 81 2 16
Up 4 0 0 79 3 14
Walk 8 1 0 73 8 10
True label
Down Jog Sit Stand Up Walk
Predicted label
0 20 40 60 80


LR:0.1, N:100, B:30
Down 21 11 0 30 34 4
Jog 3 83 1 1 11 1
Sit 0 5 94 1 0 0
Stand 0 5 0 45 50 0
Up 11 18 0 26 39 6
Walk 11 17 0 19 47 6
True label
Down Jog Sit Stand Up Walk
Predicted label
0 20 40 60 80

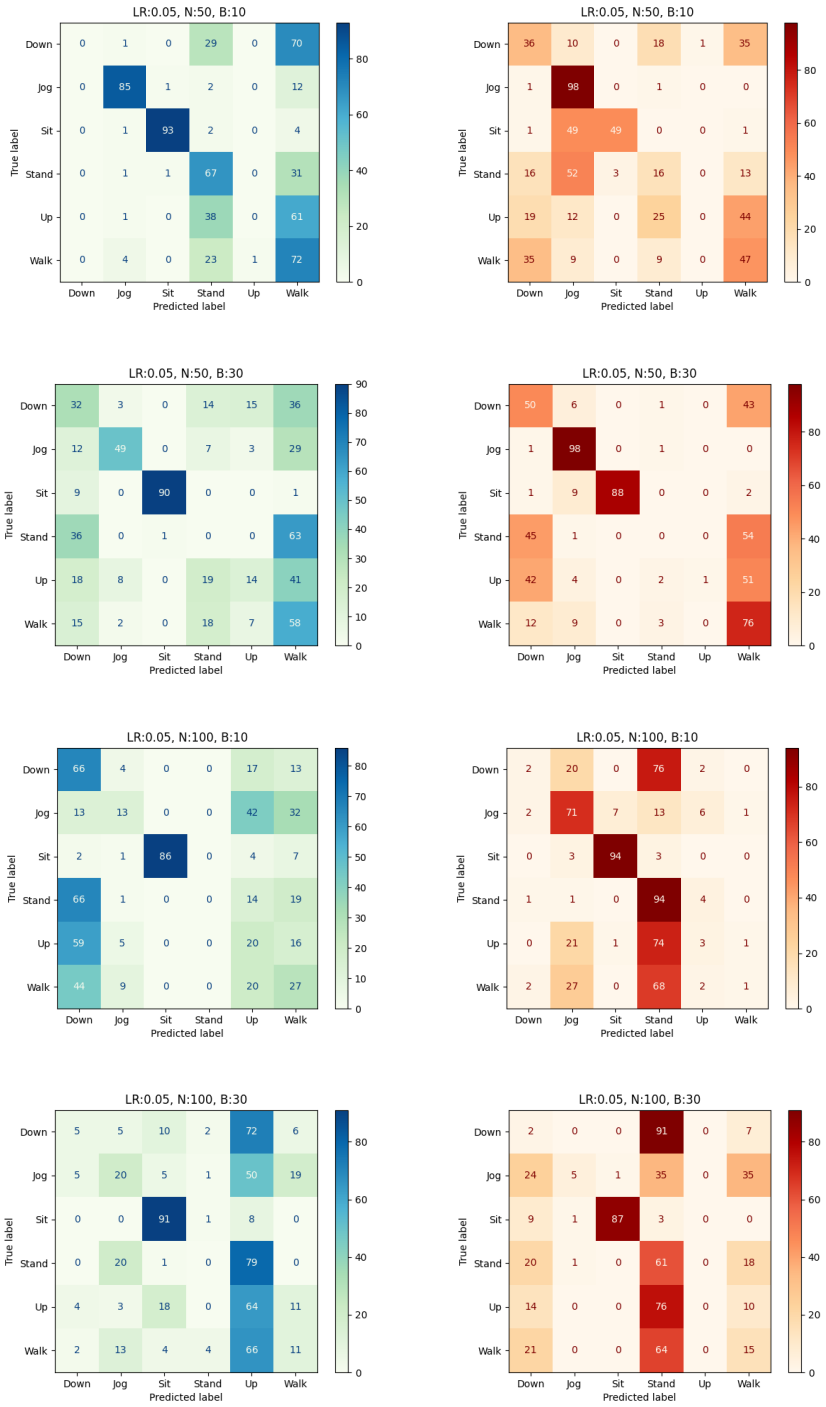


Figure 7. Confusion matrices for all trained 8 models in Part (b) (left) versus the models in Part (c) (right). The ideal case is to get a diagonal matrix.

- Notice how early stopping helps the confusion matrices to "move towards being more diagonal". This enables the models to be more generalizable, with the probable cost of a lesser accuracy score on the test set, especially on the Top-1 score.