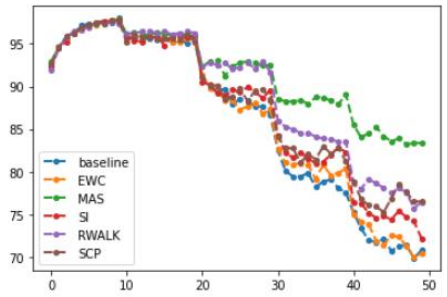
HW14 report r09921026 李育倫

1. Plot the learning curve of the metric with every method. (The Plotting function is provided in the sample code.)



2. Describe the metric.

   從 train 和 evaluate 的 function 中可得知不同方法的 acc 是用加總取平均的方式。

		Test on			
		Task 1	Task 2	.....	Task T
Rand Init.		$R_{0,1}$	$R_{0,2}$		$R_{0,T}$
After Training	Task 1	$R_{1,1}$	$R_{1,2}$		$R_{1,T}$
	Task 2	$R_{2,1}$	$R_{2,2}$		$R_{2,T}$
	⋮				
	Task T-1	$R_{T-1,1}$	$R_{T-1,2}$		$R_{T-1,T}$
	Task T	$R_{T,1}$	$R_{T,2}$		$R_{T,T}$

$$\text{Accuracy} = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}$$

3. Paste the code that you implement Omega Matrix for MAS.

```
def calculate_importance(self):
    precision_matrices = {}
    # initialize Omega(Ω) matrix (all filled zero)
    for n, p in self.params.items():
        precision_matrices[n] = p.clone().detach().fill_(0)
        for i in range(len(self.previous_guards_list)):
            if self.previous_guards_list[i]:
                precision_matrices[n] += self.previous_guards_list[i][n]

    self.model.eval()
    if self.dataloader is not None:
        num_data = len(self.dataloader)
        for data in self.dataloader:
            self.model.zero_grad()
            output = self.model(data[0].to(self.device))
            ###################################################################
            #####   TODO: generate Omega(Ω) matrix for MAS.    #####
            ###################################################################
            output.pow_(2)
            loss = torch.sum(output,dim=1)
            loss = loss.mean()
            loss.backward()
            ###################################################################
            for n, p in self.model.named_parameters():
                    precision_matrices[n].data += p.grad.abs() / num_data

        precision_matrices = {n: p for n, p in precision_matrices.items()}
    return precision_matrices
```