**Site A – 2019**

**Dataset:**

**Total Data** -> X: (2235904, 28, 6), Y: (2235904,)

**Subset Data**: 25% of total data -> X: (558976, 28, 6), Y: (558976, )

**Train Data:** 80% of subset data -> X: (447180, 28, 6), Y: (447180,)

**Val Data:** 20% of subset data -> X: (111796, 28, 6), Y: (111796,)

**Model Architecture:**

Input -> 256 -> 128 -> 64 -> 32 -> num

classes

Output

**Layers:** Bayesian Linear

**Activation:** ReLu

**Loss:** ELBO -> CE (Class Weights) + KL Divergence (Regularization term)

**Optimizer:** Adam

**Hyper-Parameters:**

BATCH\_SIZE = 64

NUM\_CLASSES = 3

INPUT\_DIM = 28 \* 6 -> Flattened dimension (T\*F)

OUTPUT\_DIM = NUM\_CLASSES

KL\_WEIGHT = 1e-6

LR = 1e-3

EPOCHS = 15

PRIORS = {

'prior\_mu': 0.0,

'prior\_sigma': 0.1,

'posterior\_mu\_initial': (0.0, 0.1),

'posterior\_rho\_initial': (-3.0, 0.1)

} (Gaussian Prior)

**Experiments**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiments** | **Train Accuracy** | **Val Accuracy** | **Kappa** | **Per Class F1** |
| prior\_sigma: 0.1 | 94% | 94.7% | 0.91 | [0.89, 0.965, 0.950] |
| prior\_sigma: 1.0 | 93.4% | 94.5% | 0.91 | [0.889, 0.965, 0.946] |
| prior\_sigma: 0.01 | 28% | 30% | 0.03 | [0.302, 0.009, 0.451] |
| prior\_mu: 1.0 | 33% | 30% | 0.03 | [0.305, 0.058, 0.426] |
| prior\_mu: -0.5 | 22.7% | 16.7% | -0.02 | [0.274, 0.036, 0.003] |

Training -> multiple outputs

Validation once at last -> multiple outputs

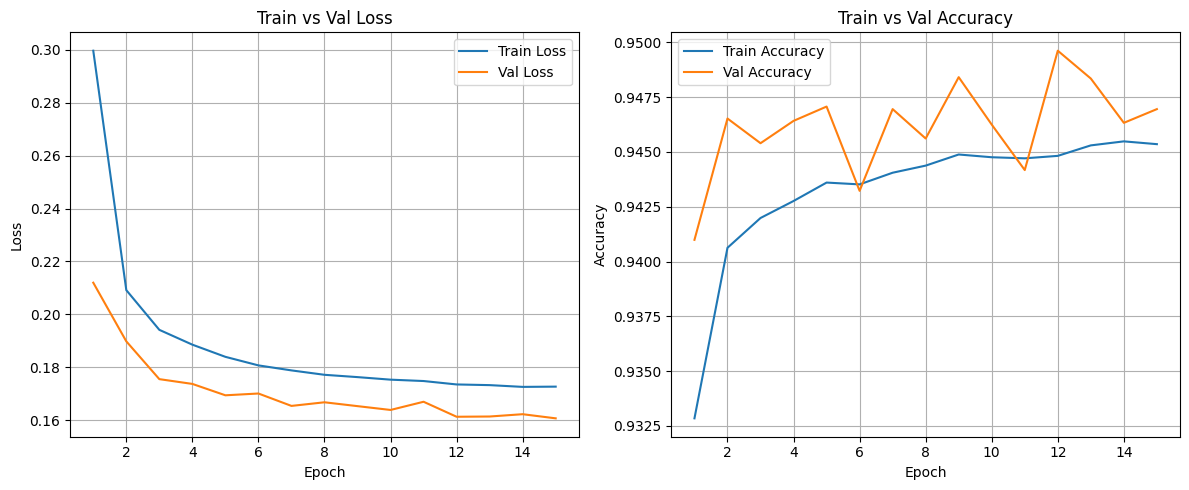
Model output -> softmax / (how loss in handling those)

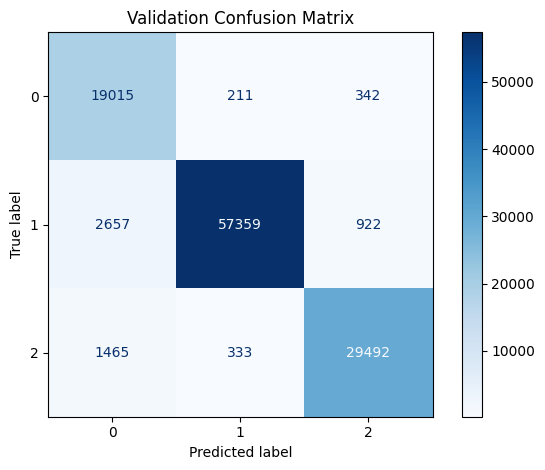
How ce works (log softmax + nll)

What is model currently generating output without softmax

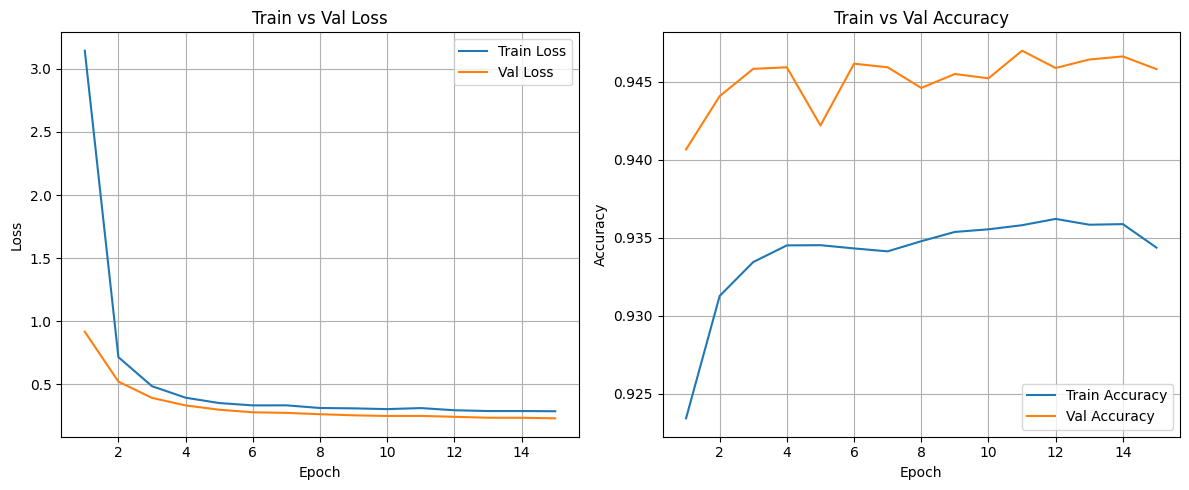
**Results:**

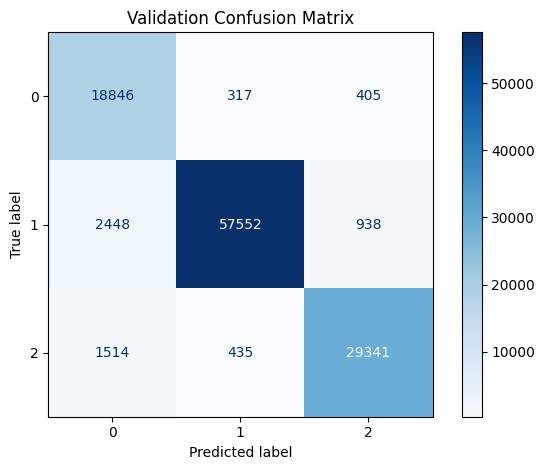
prior\_sigma: 0.1



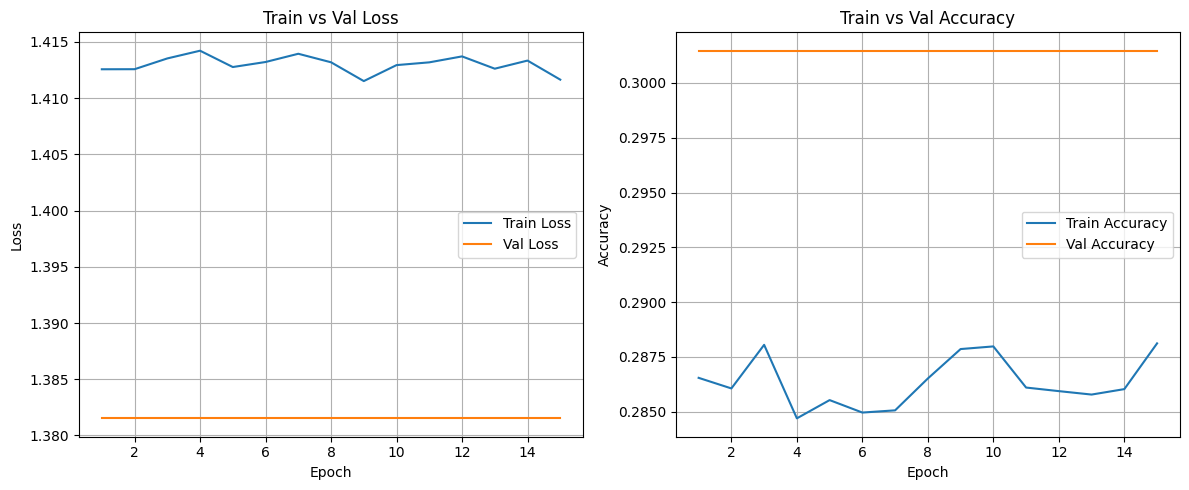


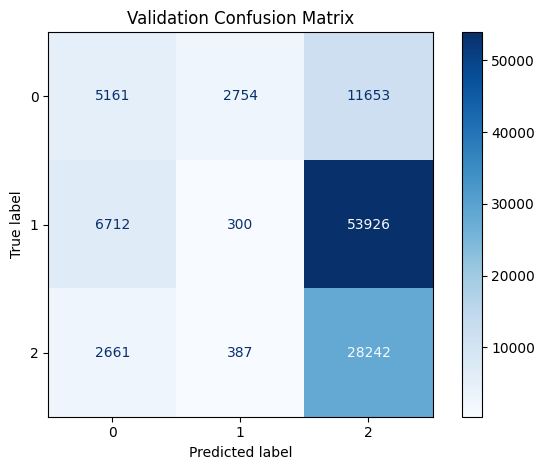
prior\_sigma: 1.0



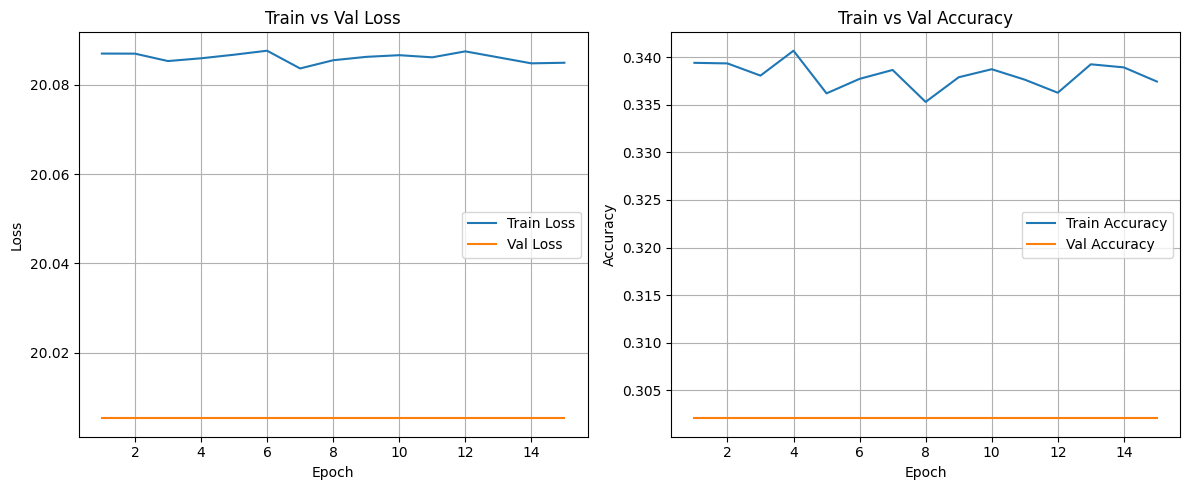


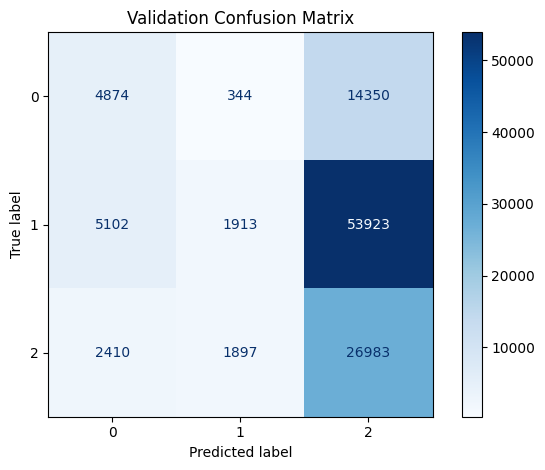
prior\_sigma: 0.01



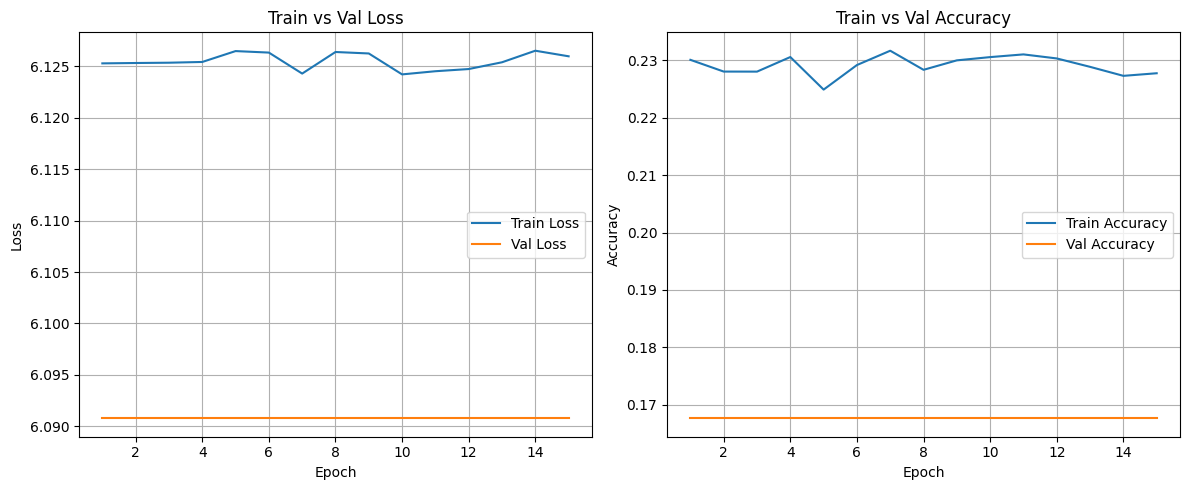


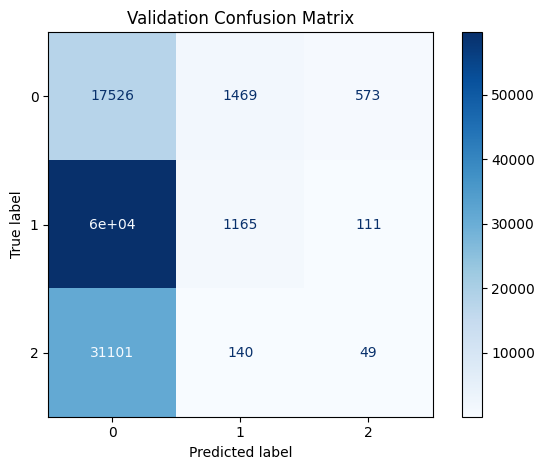
prior\_mu: 1.0

****

****

prior\_mu: -0.5

****

****

**Conclusion:** Changing the **Prior’s standard deviation** changes the prior distribution spread. This effects the regularization. If the prior is too narrow (low std), it will not allow the model learn the weights that are away from the prior and the model will be penalized. So model will underfit. If the prior is too wide (large std), the model will try to learn every weight that describes the data, i.e, posterior will be very less penalized if the model wanders off too much because prior is too wide. This could result in overfitting and the prior is no longer guiding the learning. The Bayesian nature of the model is also diminished because if the prior is too wide, the model can learn whatever weights fit the data best, and the KL regularization becomes negligible.

Changing the **Prior’s mean** changes where the prior distribution is. It tells the model where should you learn the weights and restricts the model. It could lead to underfitting since we are saying that the weights should be here and there. If model tries to learn anywhere else, it is penalized. So, the model will waste its learning capacity (epochs) in fighting the prior.

* I didn’t experiment this but while doing changes in prior, I automatically understood what the changes in initial posterior would results in. Changes in initial posterior would just change the initial place from where we start learning. So it may slow the learning and the model could get stuck in the local minima while coming close to prior. The initial posterior doesn’t determine the final performance if training is long and gradients are well-behaved — but it can significantly affect convergence speed and stability.