Project 4: Pretraining

SW10 – Jannine Meier – FS24

Preprocessing: Winogrande dataset

”I moved the couch from the garage to the backyard to create space. The _ is small.”

1. Replace the blank with each option and combine the two sequences

”… to create space. The couch is small.” + [SEP] + ”… to create space. The garage is small.”

🚫 *Stemming and Lemmatizing*

2. Tokenize the sequence using BertTokenizer

-> lowercase, add special tokens, add padding to max_length

Remove punctuation and stopwords 🚫

[CLS] i moved … to create space . the couch is small . [SEP] i moved … to create space . the garage is small . [SEP] [PAD] … [PAD]

3. Return for each sequence an input_id, attention_mask and a label

label remapping to right option

- **Input_ids**: unique ID for each token
- **Attention_mask**: 1 for tokens, 0 for padding
- **Label**: 0 if first sentence is true, 1 otherwise

Preprocessing: Anagram datasets

1. Replace the <sep> with the Bert seperator token

b p k <sep> k p b

b p k [SEP] k p b

2. Tokenize the sequence using BertTokenizer

-> add special tokens, add padding to max_length

[CLS] b p k [SEP] k p b [SEP] [PAD] … [PAD]

3. Return for each sequence an input_id, attention_mask and a label

- **Input_ids:** unique ID for each token
- **Attention_mask:** 1 for tokens, 0 for padding
- **Label:** 1 if it is an anagram, 0 otherwise

Model: Pretrained encoder transformer

BertForSequenceClassification

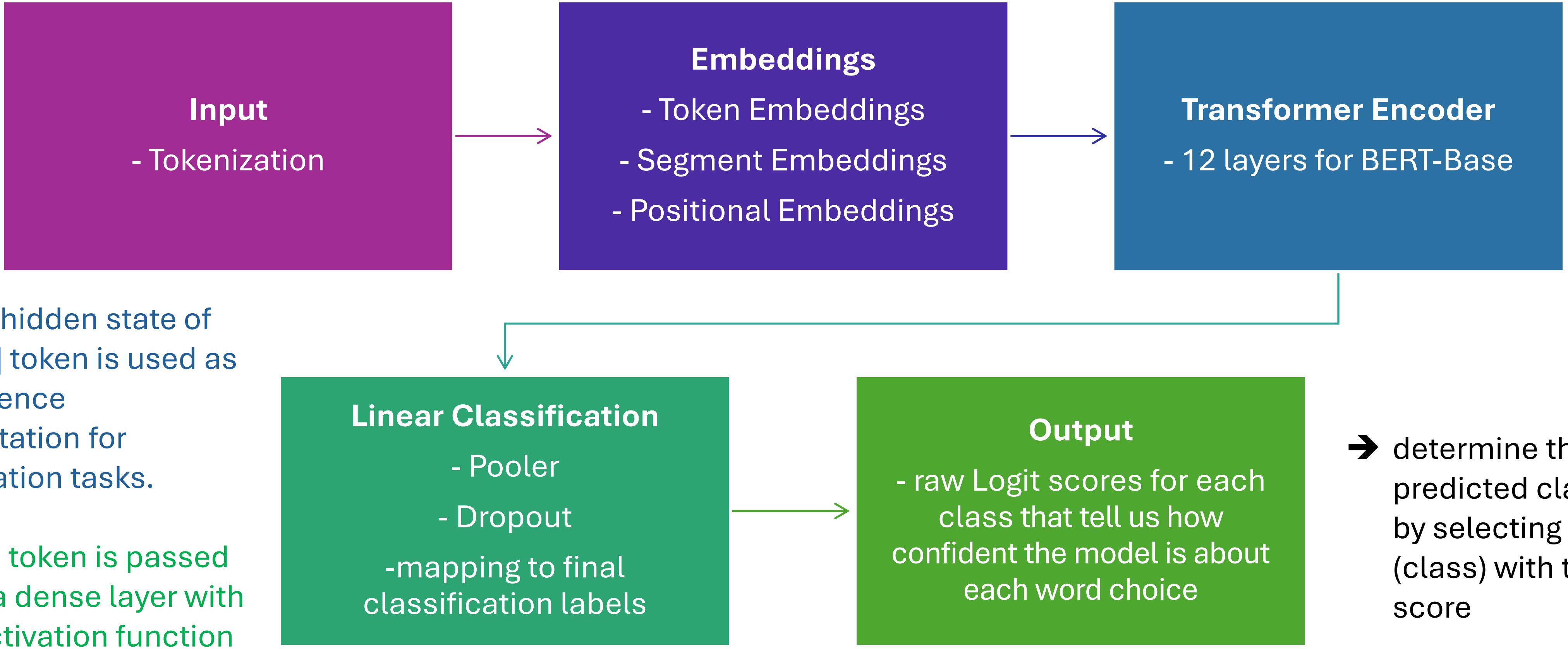
(a pre-trained BERT model with a classification layer on top)

Model Specification

- Pre-trained Knowledge
    - great language understanding
- Bidirectional Contextual Understanding
    - consider both the words before and after the target word in the sentence
- Fine-Tuning for Our Specific Need
    - classify sequences into categories
    - decide if a sentence makes sense with one word or another

Model: Pretrained encoder transformer

BertForSequenceClassification



Input

- Tokenization

Embeddings

- Token Embeddings
- Segment Embeddings
- Positional Embeddings

Transformer Encoder

- 12 layers for BERT-Base

Linear Classification

- Pooler
- Dropout

-mapping to final classification labels

Output

- raw Logit scores for each class that tell us how confident the model is about each word choice

The final hidden state of the [CLS] token is used as the sequence representation for classification tasks.

The[CLS] token is passed through a dense layer with a tanh activation function to create a pooled output.

➔ determine the predicted class label by selecting the index (class) with the highest score

Experiment settings

Sweep configurations

- method: grid
- learning_rate: 1e-6, 1e-5, 1e-4, 1e-3, 1e-2

Fixed parameters

- batch_size: 32/64 (max possible)
- num_epochs: 100/10
- warmup_ratio: 0.1
- early_stopping_patience: 20
- Loss Function: CrossEntropy
- Optimizer: AdamW with weight_decay: 0.1

Results: Winogrande

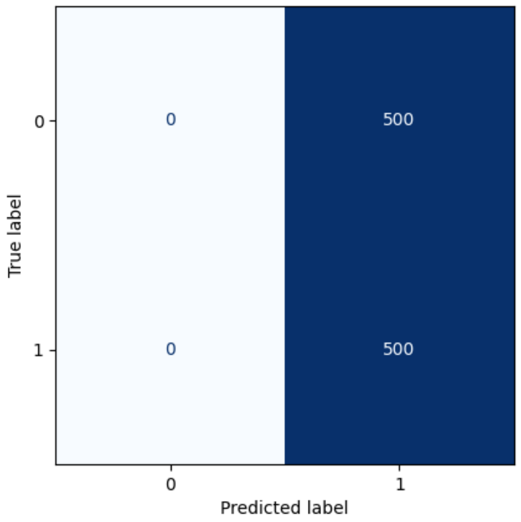
Best model

- Validation accuracy: 55.3%
  - last project 50.9%
- Test accuracy: 66.2%
  - last project 50.0%
- Small increase in accuracy
- Slightly better performance than random

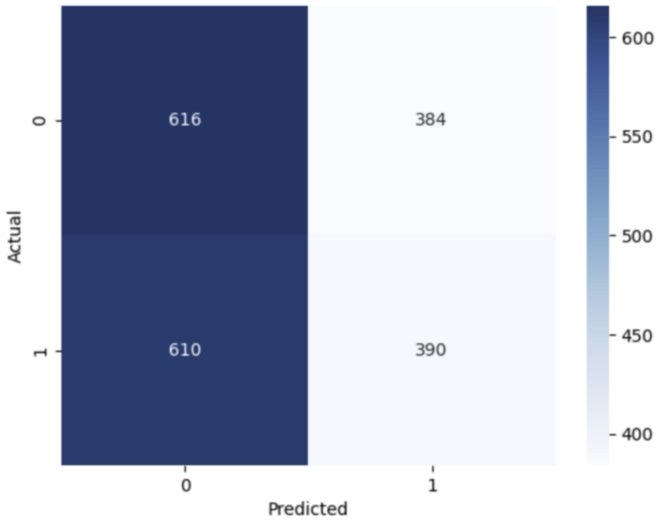
Confusion-matrix

- more balance in predictions

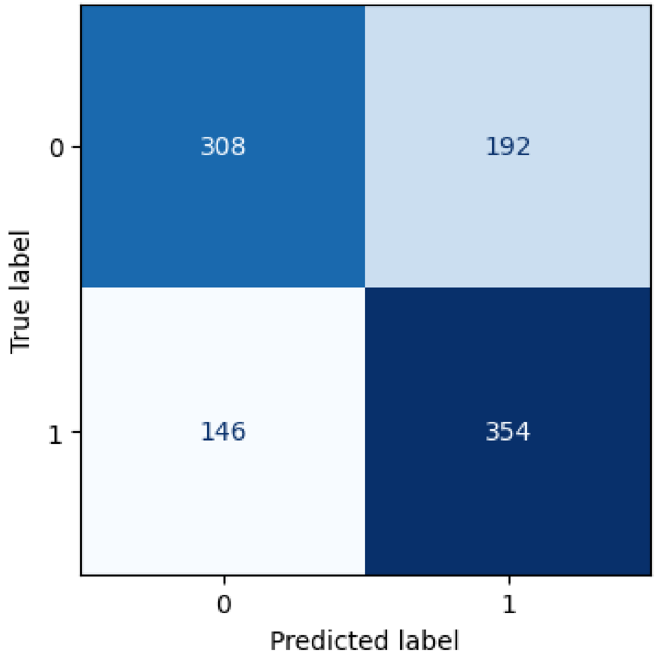
Project 2 RNNs



Project 3 Transformers

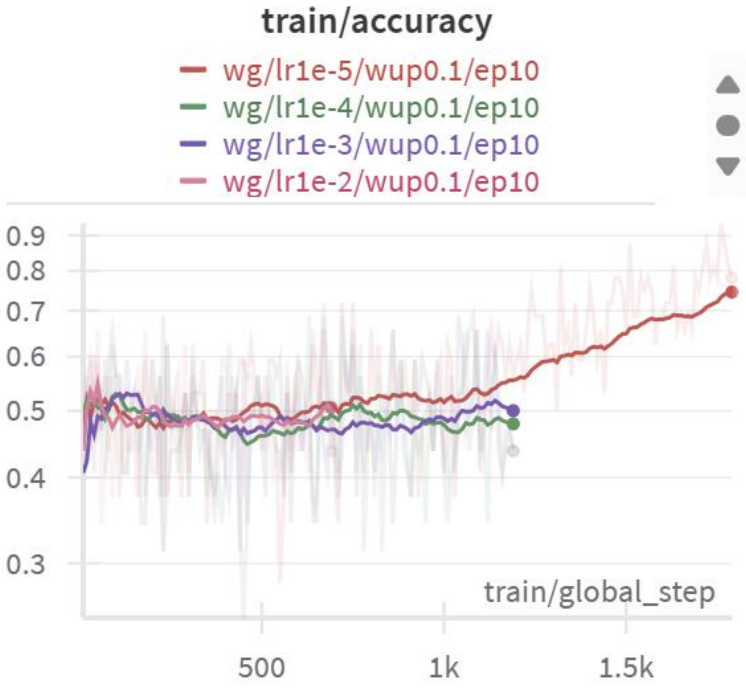


Project 4 Pretraining

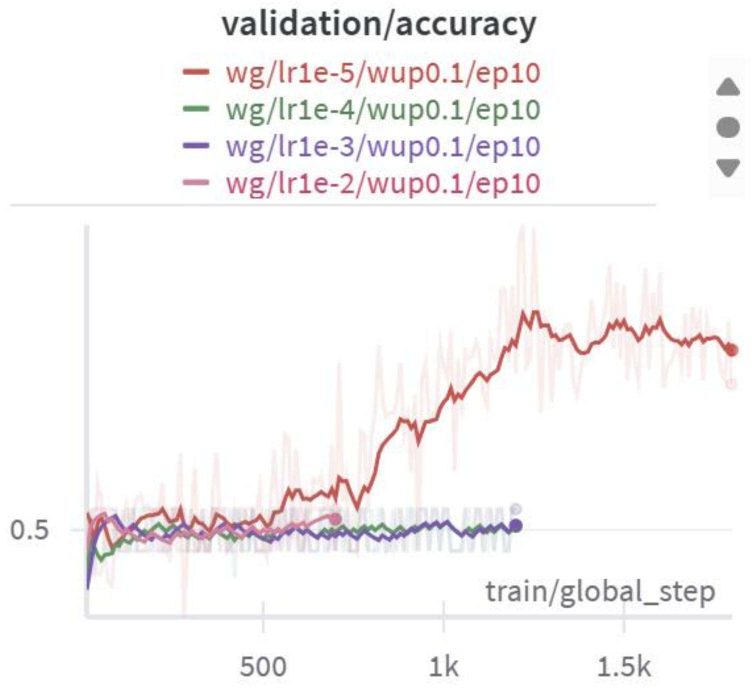


Results: Winogrande

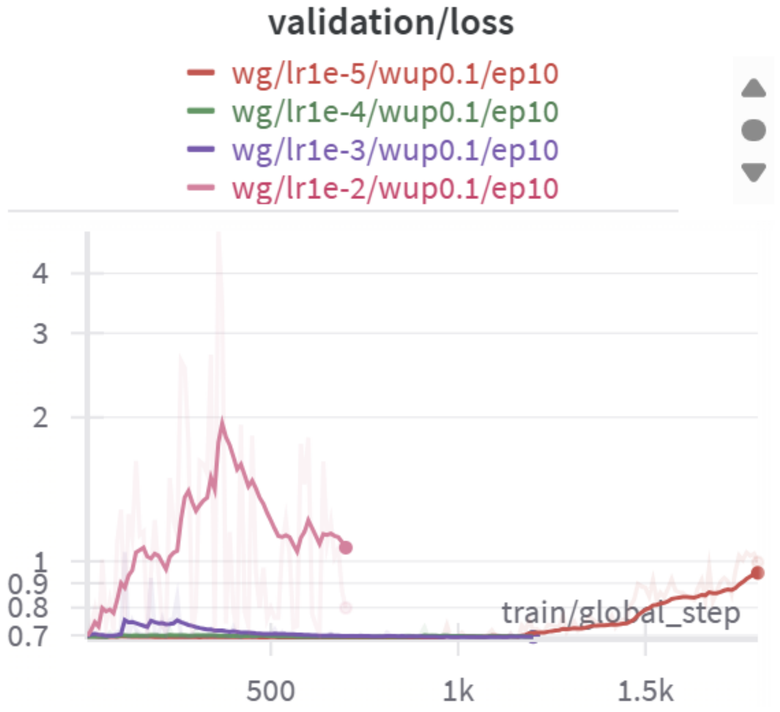
Finally some learning progress in train and validation!



train/accuracy



validation/accuracy



validation/loss

Results: Anagram Small

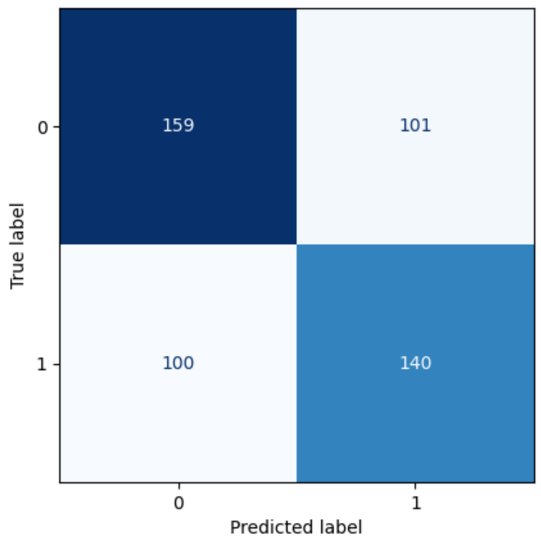
Best model

- Validation accuracy: 95.6%
  - last project 56.4%
- Test accuracy: 96.4%
  - last project 53.6%

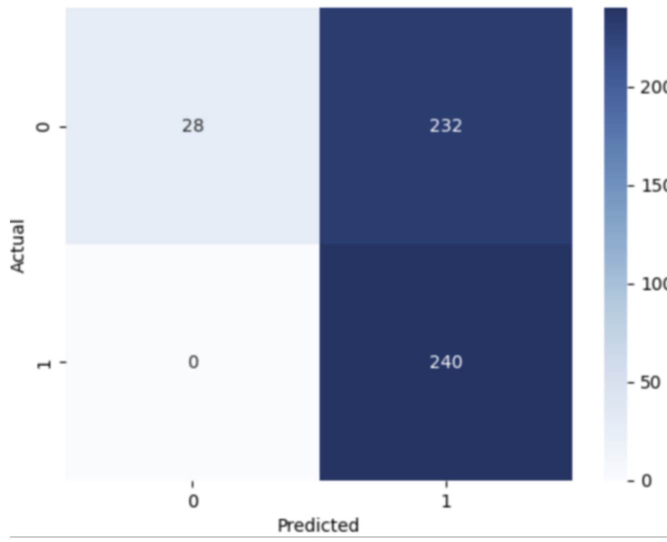
Confusion-matrix

- more balance in predictions
- Performs much better than previous projects

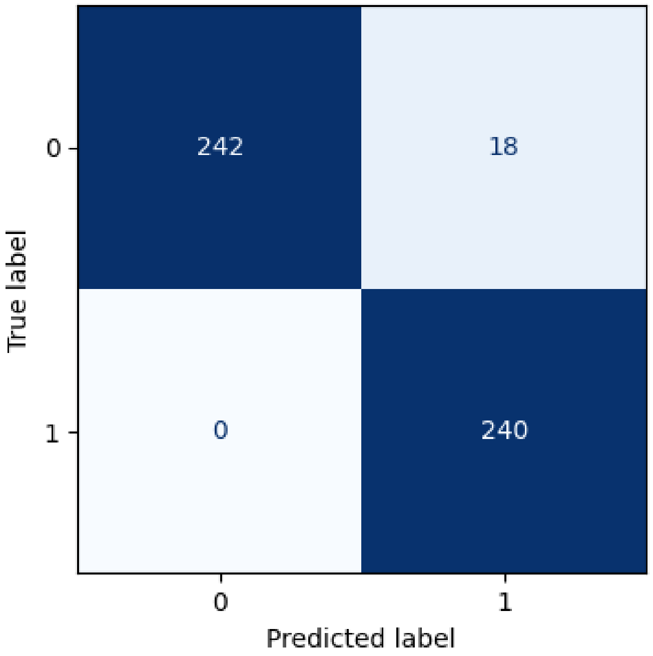
Project 2 RNNs



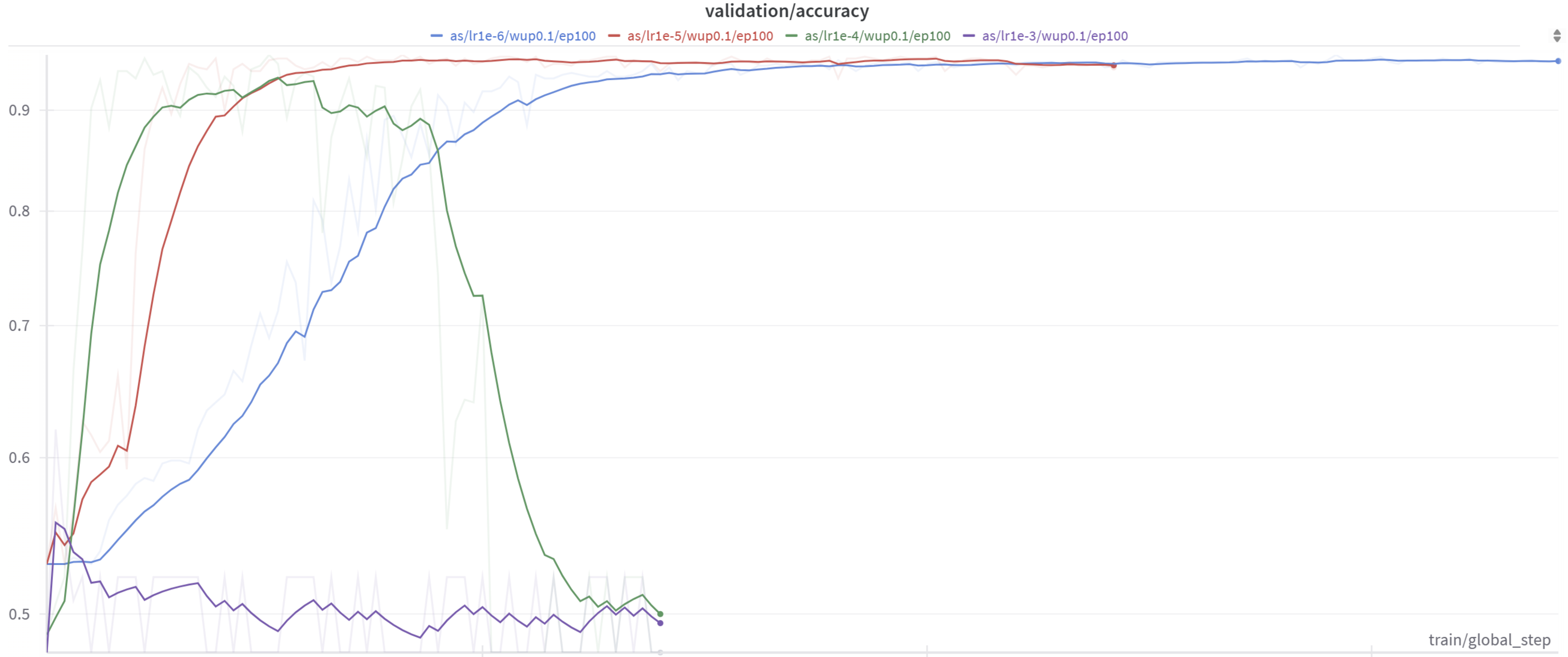
Project 3 Transformers



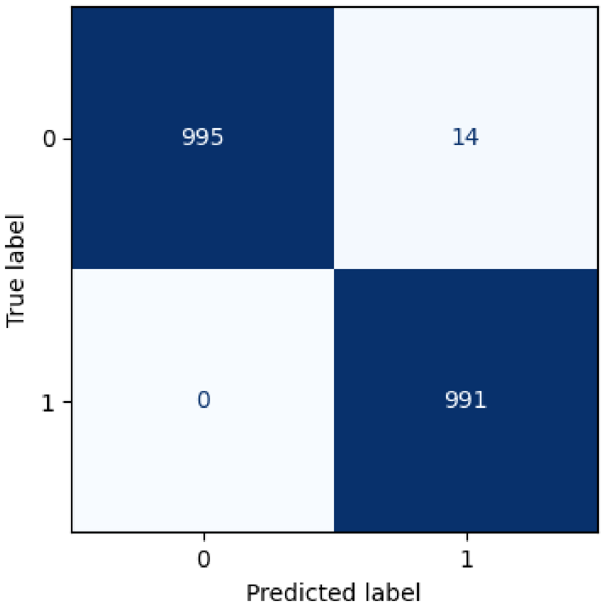
Project 4 Pretraining



Results: Anagram Small

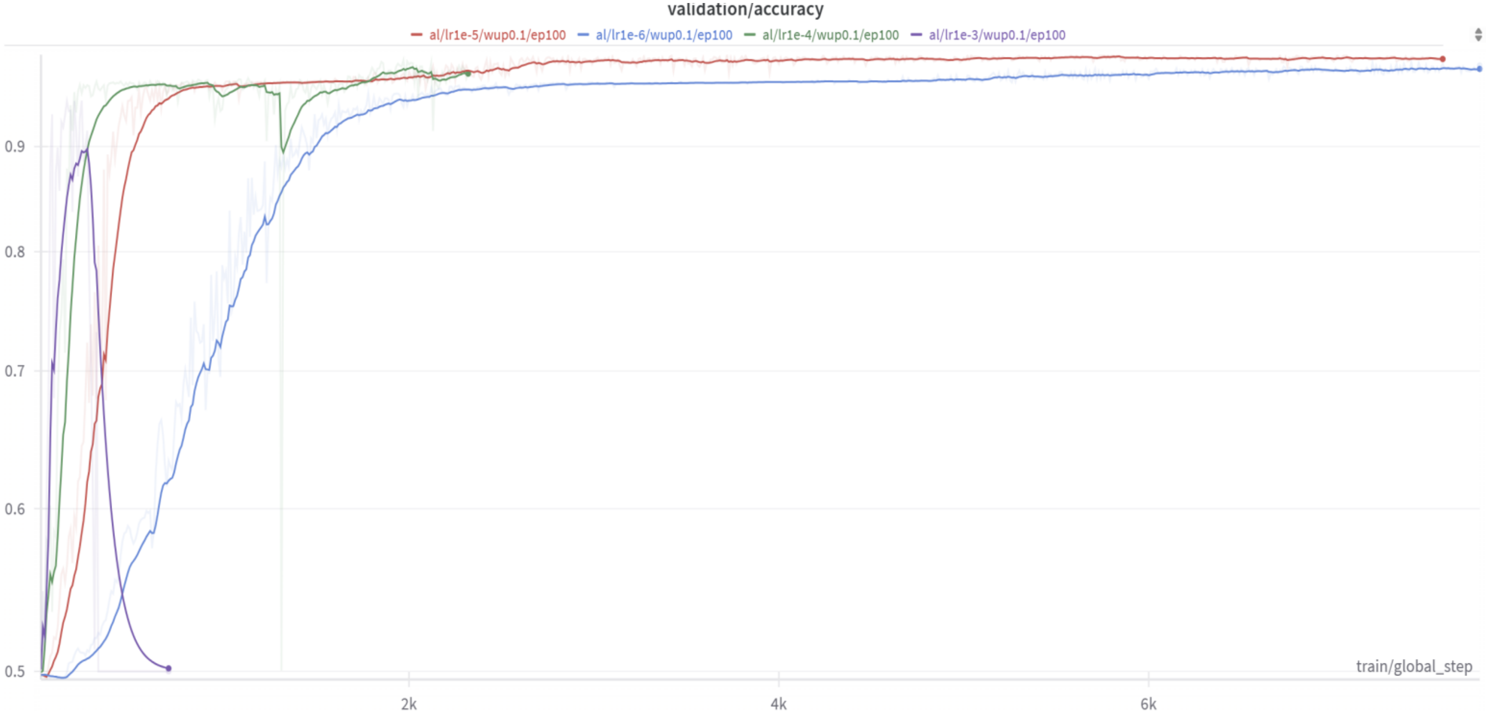


Results:
Anagram Large



Best model

- Validation accuracy: 99.5%
  - last project 99.9%
- Test accuracy: 99.3%
  - last project 97.2%



Confusion-matrix

- Almost perfect performance

Conclusions

Interpretation

Winogrande dataset

- still too complex for my model choice

Anagram datasets

- almost perfect performance

➔For all tasks small learning rates (1e-5 and 1e-6) showed best results

Lessons Learned

Less Data required

- with the pretrained transformer I already got good results on the smaller dataset as well

Warmup learning rate

- try fixed steps and ratio