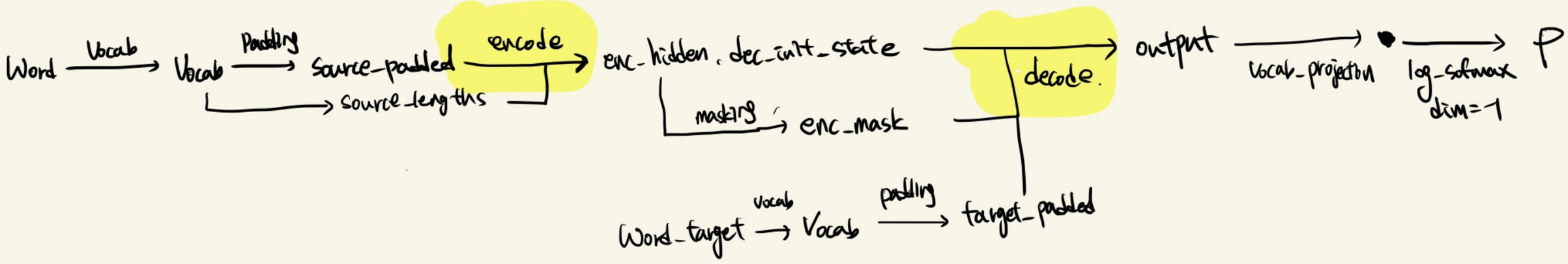
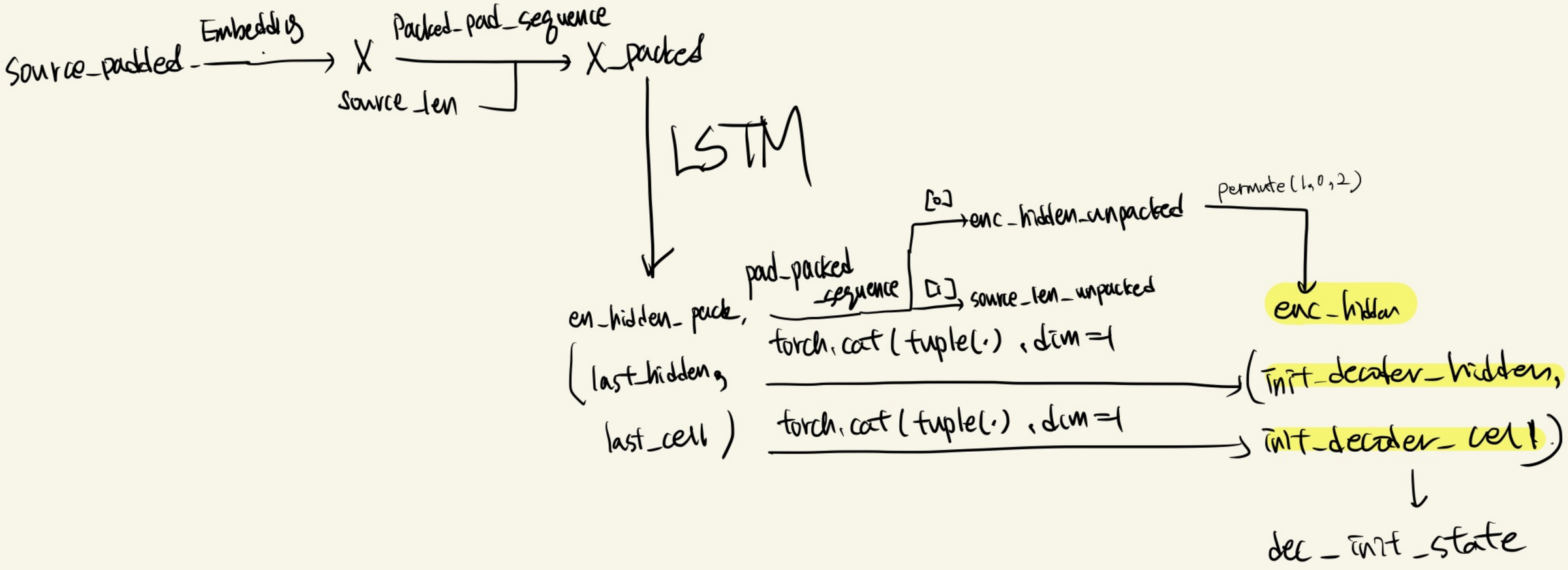
CS 224N - A4, nmt machine

Entire - code flow



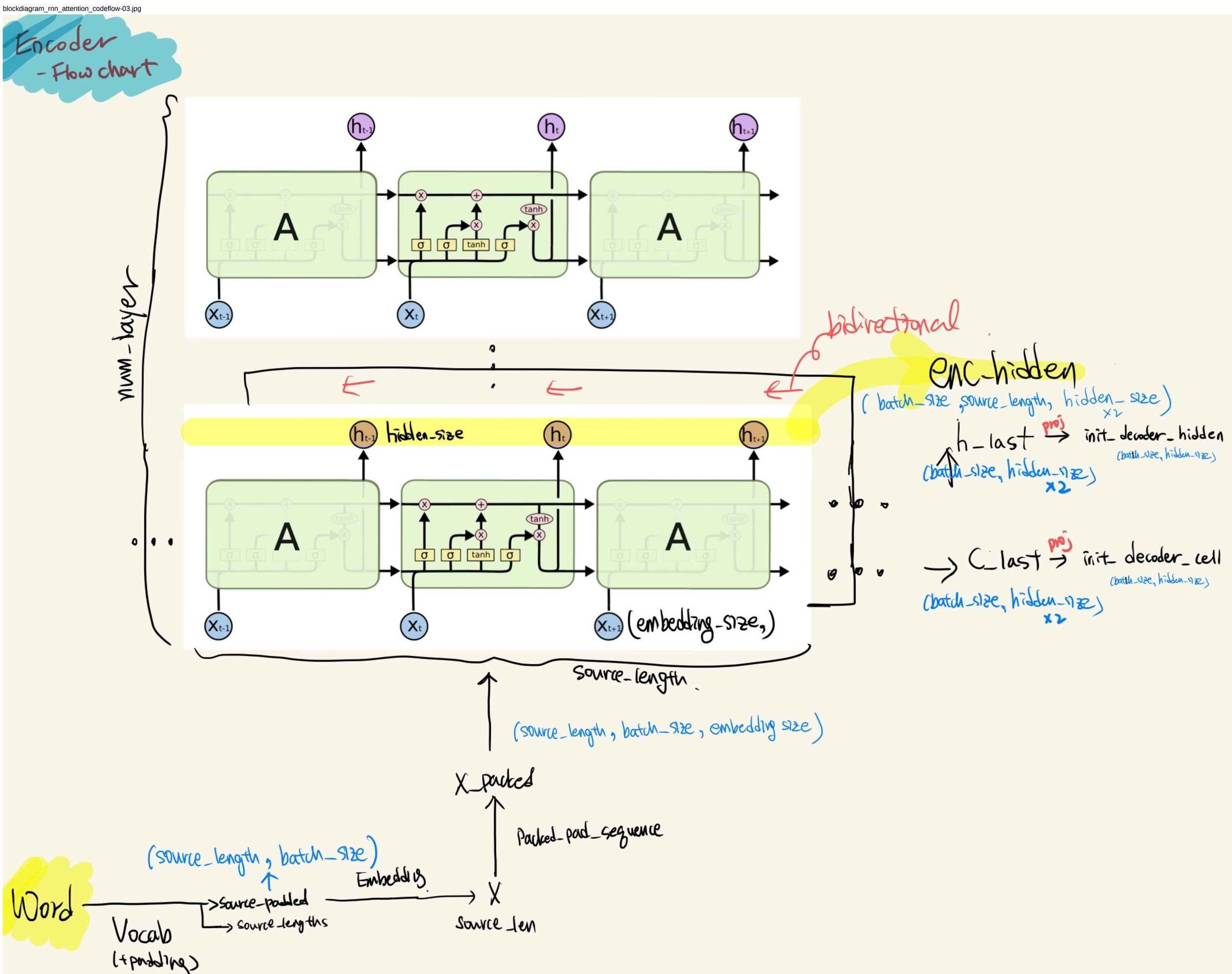
blockdiagram_rnn_attention_codeflow-02.jpg

Encoder
-code flow



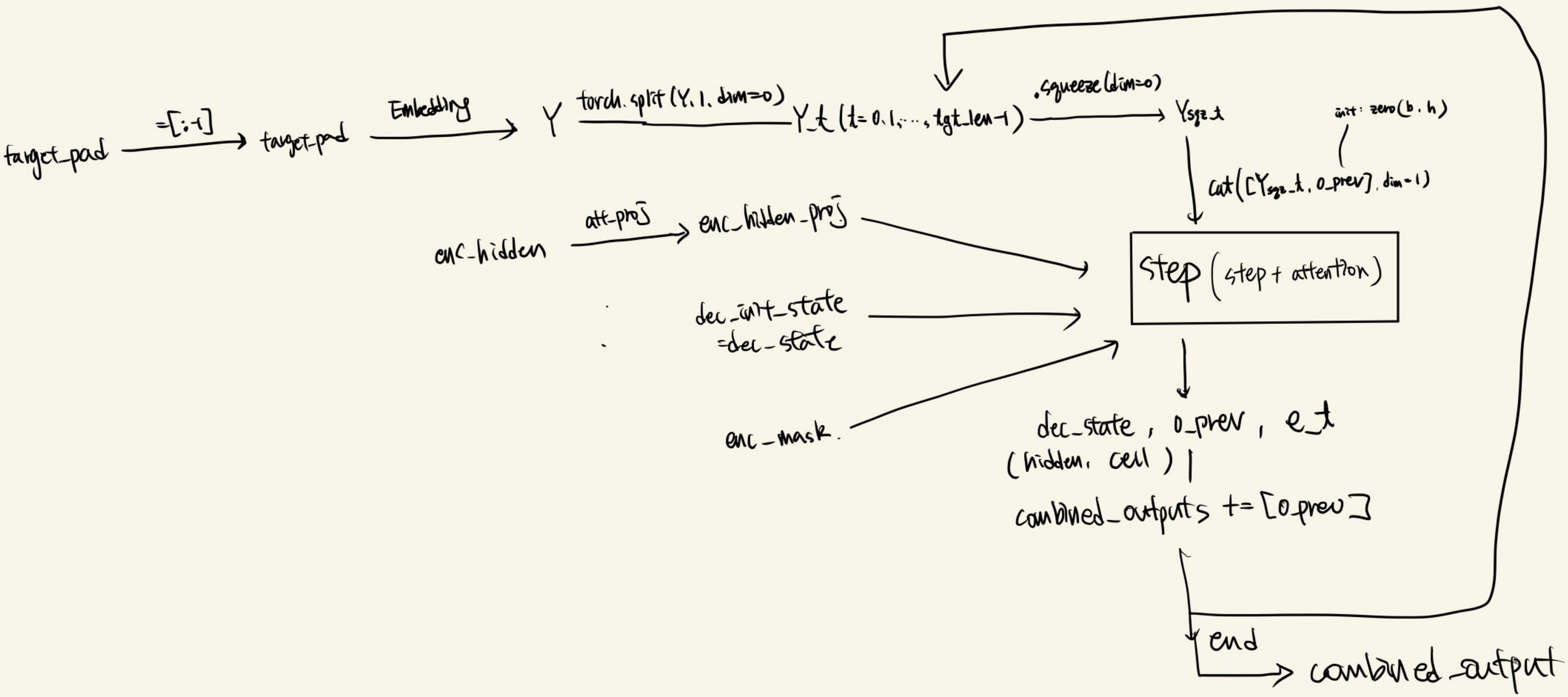
blockdiagram_rnn_attention_codeflow-03.jpg

Encoder - Flow chart



blockdiagram_rnn_attention_codeflow-04.jpg

Decoder
- code flow



blockdiagram_rnn_attention_codeflow-05.jpg

Decoder - Flowchart

Shape - word
src_len : source_length
b : batch_size
e : embedding_size
h : hidden_size
tgt_len : target_length

Attention

(b, src_len, 2h) → hmm → (b,1,2h) → squeeze → a_t (b,2h)

α_t (b, src_len) → unsqueeze → (b,1,src_len)

softmax → (b, src_len, 1)

enc_hidden_proj (b, src_len, h) → bmm ← (b,h,1) ← unsqueeze

proj

step

concat → U_t (b,2h) —proj→ V_t (b,h) → Activation (ex. tanh) → dropout → Out_t (b,h)

Out_t → torch.stack [out_0, ...] → Out (tgt_len, b, h)

Update to out_prev

t = 0, 1 ... tgt_len - 1

(h_0 ... h_n) = enc_hidden (b, src_len, h×2) — pf. encoder

dec_hidden (b, h)

step

attention ← dec_hidden ...

encoder

bi-direction model

source_length

(source_length, batch_size, embedding size)

X_packed

Decoder

uni directional - model

pf. encoder → init_decoder_hidden

pf. encoder → init_decoder_cell

LSTM LSTM ... LSTM

(b, e)

dec_hidden (both)

LSTM — dec_cell (b,h) — LSTM ...

(zeros) out_prev (b,h) → cat([ ], dim=1)

cat ...

out t=0 (b,h)

Y_sqz_t (b, e)

squeeze

squeeze

Y_t (t = 0, 1, ..., tgt_len-1) (1, b, e)

torch.split (Y, 1, dim=0)

Y (tgt_len, b, e)

Embedding

target_padded (tgt_len, b, vocab) —[:-1]→ target_pad (tgt_len, b, vocab)

word → Vocab

blockdiagram_rnn_attention_codeflow-06.jpg

evaluation
- code + flow chart

out
(tgt_len, b, h)
↓
vocab_projection
(tgt_len, b, vocab_size)
↓
log_softmax
(tgt_len, b, vocab_size)
↓
P
↓

evaluate
(masking)

torch.gather(P, index= target_padded[1:].unsqueeze(-1), dim=-1).squeeze(-1)
(tgt_len, b, 1)
↓
(tgt_len, b)
↓

* ← target_masked ← target_padded = self.vocab.tgt['<pad>'].float()
(tgt_len, b)
↓

target_gold_words_log_prob
(tgt_len, b)
↓

• sum(d=0)
↓

Scores
(b,)