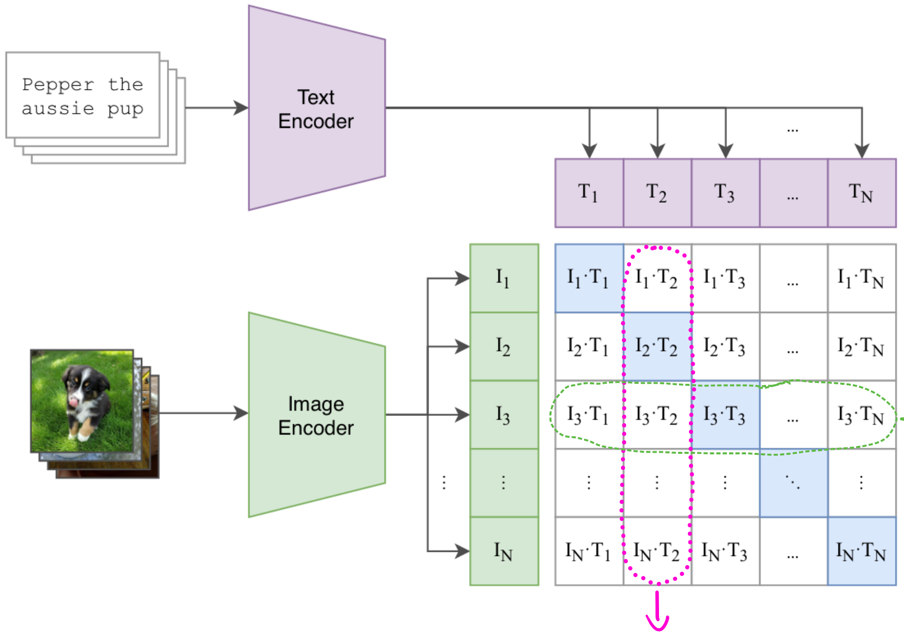
From CLIP to SigLip

CLIP stands for Contrastive Language-Image Pretraining.

What is contrastive learning?

(1) Contrastive pre-training



	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$	...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$	...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$	...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	⋱	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$	...	$I_N \cdot T_N$

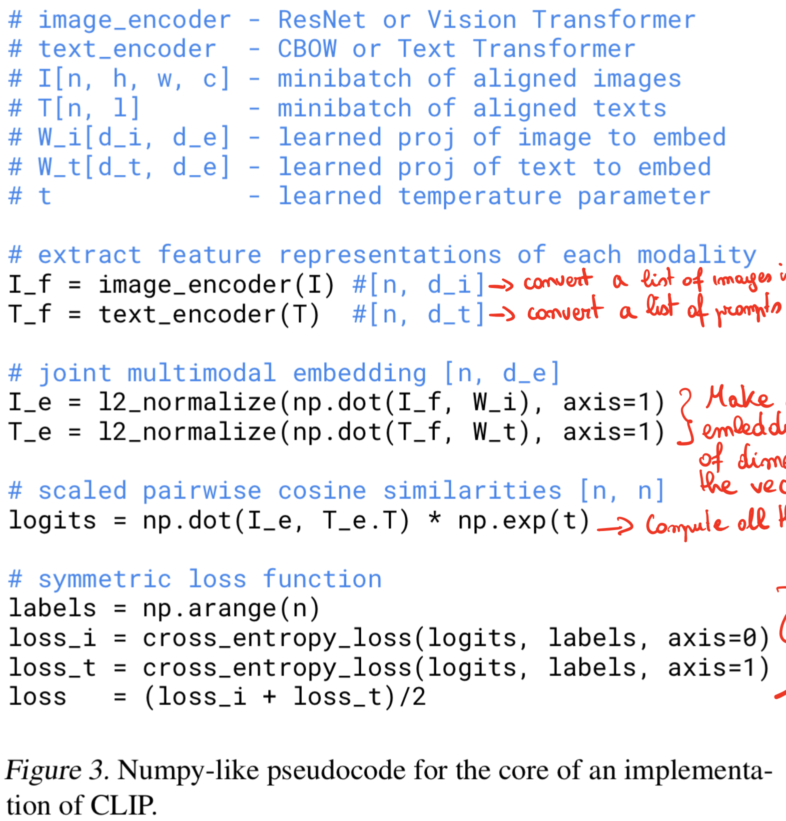
→ we want the third item to be the maximum in this row

we want the second item to be the maximum in this column

Problem: how do we tell the model we want one item in each row/column to be maximized while minimizing all the others?

Hint : This is very similar to language modeling in which we want a single token to be the next one given the prompt...

Solution : We use the Cross-Entropy Loss!



```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

-> convert a list of images into a list of embeddings

-> convert a list of prompts into a list of embeddings

Make sure both image and text embeddings have the same number of dimensions, and then normalize the vectors

-> Compute all the possible dot products.

Teach the model which item in each row/column needs to be maximized

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

Numerical stability of the softmax

$\forall i \in 1 \dots N \qquad S_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$

The softmax makes all the elements of a vector in such a way that they're in the real range $[0,1]$ and they sum up to 1.

Problem: the softmax is numerically unstable, as the exp function can grow fast and may not fit in a 32 bit floating-point number.

Solution: do not make the exp grow to infinity.

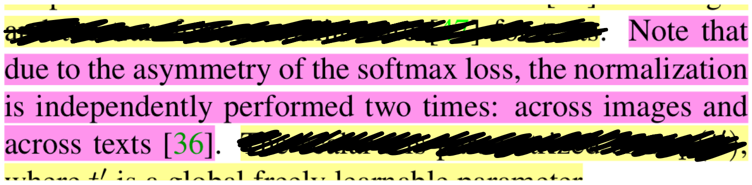
$$S_i = \frac{c \cdot e^{a_i}}{c \cdot \sum_{k=1}^{N} e^{a_k}} = \frac{e^{\log(c)} e^{a_i}}{e^{\log(c)} \sum_{k=1}^{N} e^{a_k}} = \frac{e^{a_i + \log(c)}}{\sum_{k=1}^{N} e^{a_k + \log(c)}}$$

We normally choose $\log(c) = -\max_i (a_i)$

This will push the arguments of the exp towards negative numbers and the exp itself towards zero.

The normalization factor in the softmax

To calculate the normalization factor, we must go through all the elements of each row and each column.

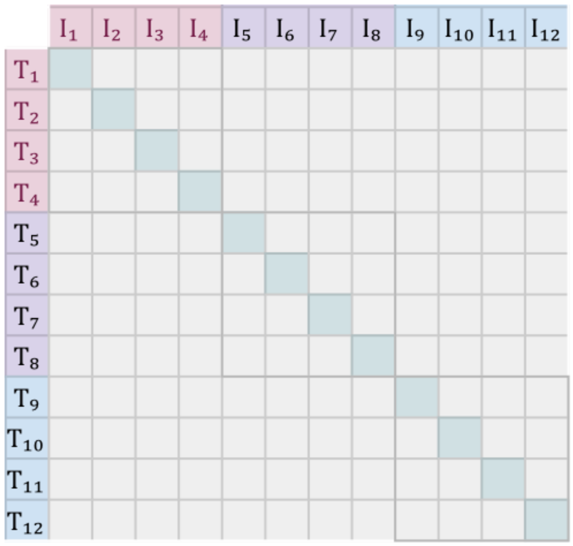


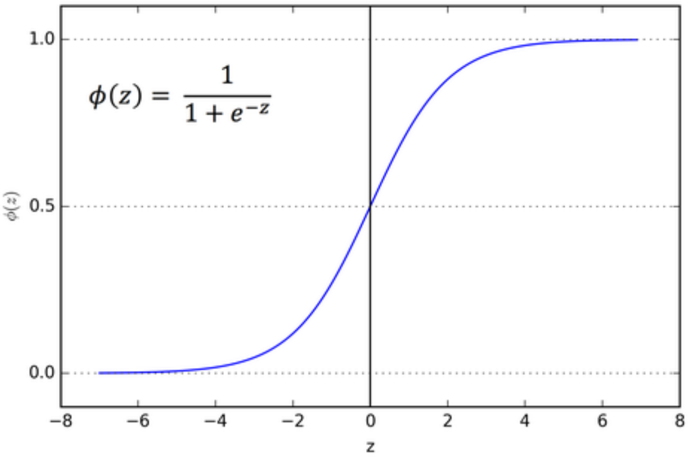
[redacted] Note that due to the asymmetry of the softmax loss, the normalization is independently performed two times: across images and across texts [36]. [redacted] where t' is a global freely learnable parameter.

$$-\frac{1}{2|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\left(\overbrace{\log\frac{e^{t\mathbf{x}_i\cdot\mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|}e^{t\mathbf{x}_i\cdot\mathbf{y}_j}}}^{\text{image}\rightarrow\text{text softmax}}+\overbrace{\log\frac{e^{t\mathbf{x}_i\cdot\mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|}e^{t\mathbf{x}_j\cdot\mathbf{y}_i}}}^{\text{text}\rightarrow\text{image softmax}}\right)$$

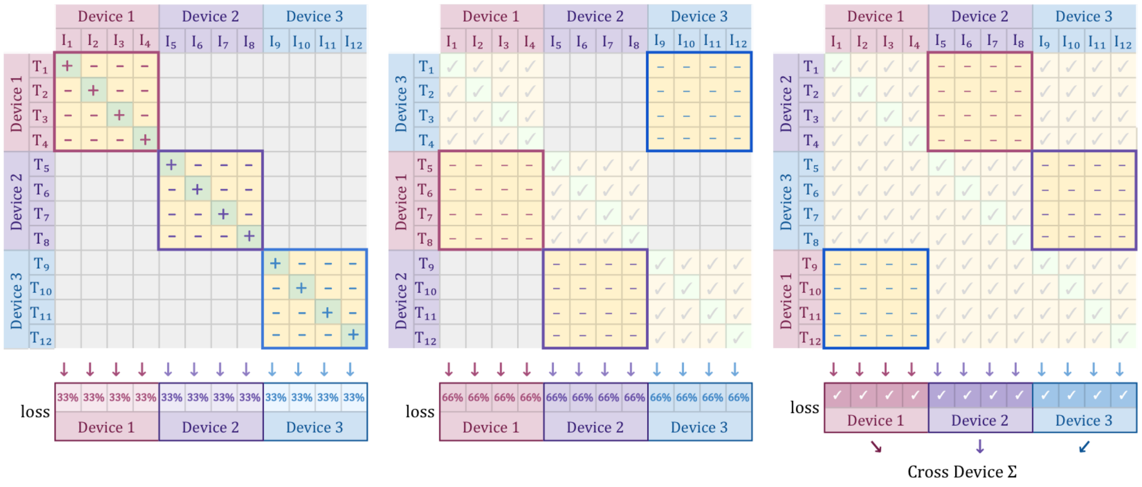
The solution is to use ... a Sigmoid!

$$-\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\underbrace{\log\frac{1}{1+e^{z_{ij}(-t\mathbf{x}_i\cdot\mathbf{y}_j+b)}}}_{\mathcal{L}_{ij}}$$





Parallel computation


Device 1
Device 2
Device 3
I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 I12
T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12
+
−
✓
loss
33%
66%
Cross Device Σ