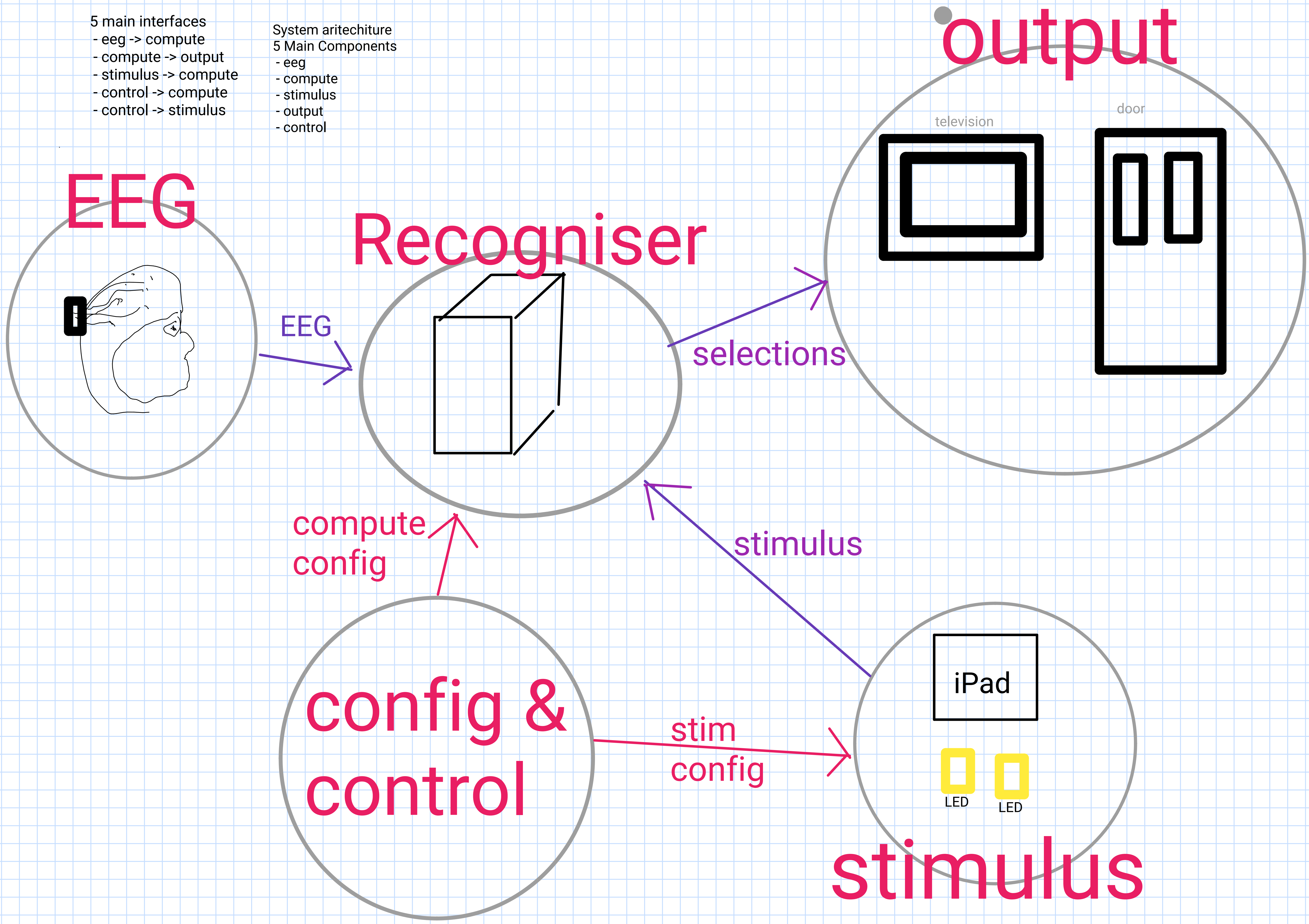
system architecture / Component Roles



Component Basic Roles & Responsibilities

EEG:
Responsibilities: acquistation of brain measurements
Inputs:
Outputs: time-stamped brain measurements over multiple channels

Stimulus:
Responsibilities: presentation of BCI required stimulus to the user, e.g. speller stiimulus
Inputs: Stimulus control messages from the controller
Outputs: time-stamped information about the current stimulus displayed to the user

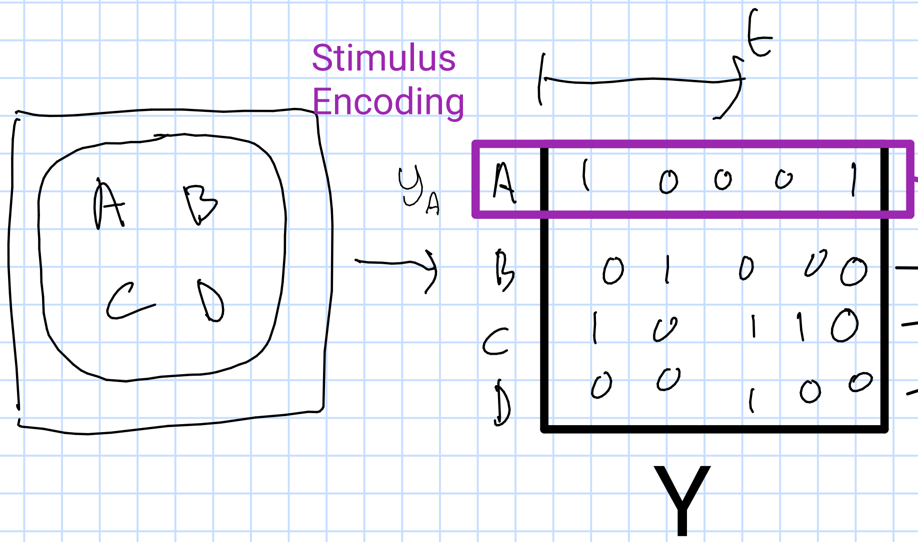
Output:
Responsibilities: translation of BCI derived selections into control of output devices, e.g. open-door, add letter to sentence.
Inputs: target selections from the Recogniser, configuration from the controller
Outputs: (whatever the output device needs)

Recogniser:
Responsibilities: translation of the EEG + stimulus information into target selections for output.
Inputs: time-stamped EEG, time-stamped stimulus information, mode control information
Outputs: target/output selections (with confidence)

Config & Control:
Responsibilities: configuration and control of the other components,
e.g. setting the stimulus rate/code, switching from calibration to online modes.
System monitoring and error logging / user information.
Inputs: saved config files
Outputs: config and control messages to the other components.

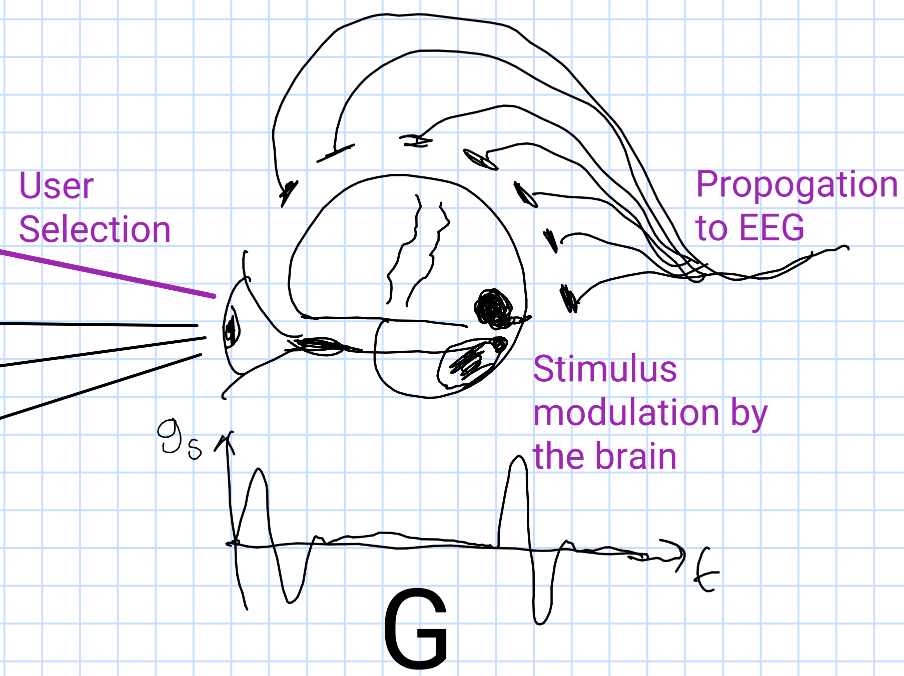
Recogniser: Forward Model

Stimulus

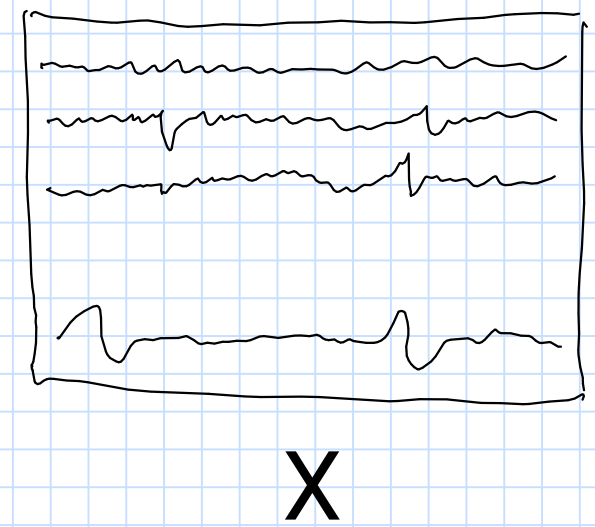


Ya - [1 x #Tr] stimulus sequence for output A
Y - [#Y x #Tr] codebook for all #Y outputs

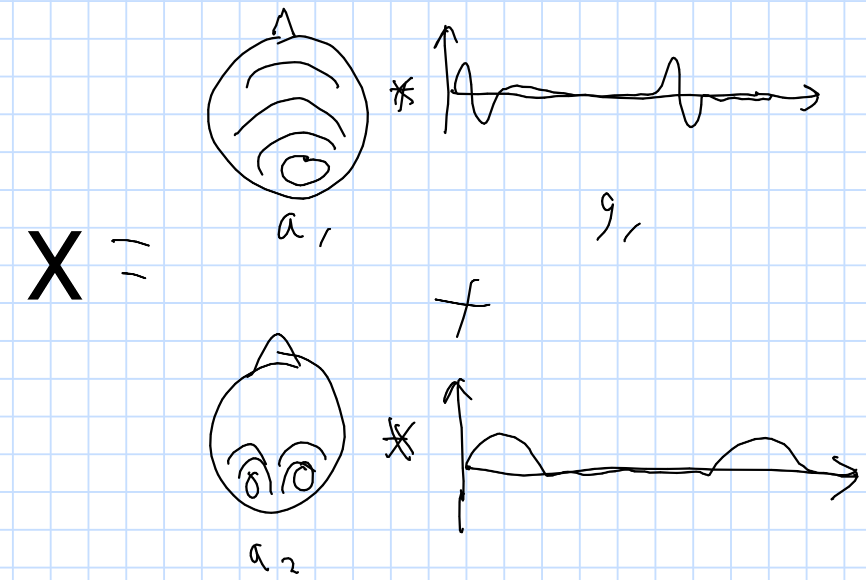
Brain Response



Measured EEG



$$X = \sum_s a_s g_s$$ source act -> EEG measure



$$G_{sa} = \sum_e m_{eA} \circledast r_{es}$$ evoked events -> source act

||

m_eA - [1 x T] brain stimulus events of type e (assuming target A)

$\circledast$ convolution

r_es - [1 x tau] impulse response for brain stimulus events of type e for source s

Note:

$$m_{eA} = E_e(y_A)$$ stim prop-> brain evoked events
i.e. the brain responds to a transformed version of the stimuli. (Modeled by Ee)

$$g_{sA} = M_{eA} r_{es}$$ Alternatively, can write the convolution with a 'structure matrix' M-[T x tau x #E x #Y]

$$X_{dT} = \sum_s a_s \sum_e m_e \circledast r_{es} + \epsilon_{dT}$$
$$= A_{ds} (M_{t\tau e} \; r_{\tau es})^T + \epsilon_{dT}$$
$$= A_{ds} \; r_{s(\tau e)} \; M_{(\tau e)T} + \epsilon_{dT}$$

N.B. assume me is from the 'true' selected target

Model Fitting by Canonical Correlation Analysis

$$Q_{st} = r_{\tau se} * Y_{te} + \epsilon_{st}$$
$$X_{dt} = a_{ds} Q_{st} + \epsilon_{dt}$$

The forward model can be summarized as:

Gs = \sum_e Me r_es $= \sum_e M_e r_{eo} = \sum_e m_e \otimes r_{es} = MR$ (modulation)

X = \sum_s a_s Gs $= \sum_o a_o G_s = AG$ (propagation)

Assuming A can be inverted, we can reverse the propogation equation to give:

Gs = pinv(A) X $= A^{\ddagger} X = w^T X =$ WX (inverse propagation)

where, w is a [d x #S] spatial filter matrix for each source

These two equations give two estimates for Gs (the neural source activation).
By requiring these estimates to be maximally correlated we obtain the CCA method for fitting the model parameters.

J_cca = \max_{w,r} correlation(MR, WX) $= \max_{w,r} \mathrm{corr}(MR, wX)$

= \max_{w,r} w'XMr / \sqrt(w'XX'w r'M'Mr) $= \max_{w,r} \frac{w^T X M r}{\sqrt{w^T X X^T w \; r^T M^T M r}}$

Rewriting for notational convience:
Cxx = XX'
Cxy = XM
Cyy = M'M

We get:

= \max_{r,w} wC_xyr / \sqrt(w'C_xx w r' C_yy r) $= \max_{w,r} \frac{w^T C_{xy} r}{\sqrt{w^T C_{xx} w \; r^T C_{yy} r}}$

There are many ways to find the maximum. On way is as follows:

Subistute: $\tilde{w} = w\, C_{xx}^{\frac{1}{2}}$ $\quad \tilde{r} = C_{yy}^{\frac{1}{2}} r$

$\Rightarrow w = C_{xx}^{-\frac{1}{2}} \tilde{w}$ $\quad r = C_{yy}^{-\frac{1}{2}} \tilde{r}$

Which gives:

$$J_{CCA} = \max_{\tilde{w},\tilde{r}} \frac{\tilde{w}^T C_{xx}^{-\frac{1}{2}} C_{xy} C_{yy}^{-\frac{1}{2}} \tilde{r}}{\sqrt{\tilde{w}^T \tilde{w} \; \tilde{r}^T \tilde{r}}}$$

$$= \max_{\tilde{w},\tilde{r}} \frac{\tilde{w}^T \tilde{C}_{xy} \tilde{r}}{\sqrt{\tilde{w}^T \tilde{w} \; \tilde{r}^T \tilde{r}}} \qquad \tilde{C}_{xy} = C_{xx}^{-\frac{1}{2}} C_{xy} C_{yy}^{-\frac{1}{2}}$$

This now has the form of a Rayleigh coefficient. For which the solution can be found by SVD

$$[\tilde{W}, \tilde{\sigma}, \tilde{R}] = \mathrm{SVD}(\tilde{X}\tilde{M})$$

Back subistuting, we get the solutions to the orginal problem:

$$w = C_{xx}^{-\frac{1}{2}} \tilde{W},$$
$$r = C_{yy}^{-\frac{1}{2}} \tilde{r}$$

Thus to solve the CCA problem we *only* need to compute and store the summary statistics:

$C_{xy} = XM$ = [d X Tau X #E x #M]

$C_{xx} = XX^T$ = [d x d]

$C_{yy} = M^T M$ = [(tau * #E) x (tau * #E) x #M]

Note:
Cxy = this is basically the the average response for each event type, i.e the event ERP
Cxx = this is the spatial covariance of the sensors
Cyy = this is the temporal auto-correlation of the brain event sequences.

Output Selection by model scoring

Given a fitted model (i.e. w,r) and #Y possible outputs
how do we identify the users selected target?

Again, many possibilities, simpilest follows directly from the two estimates for the source activation (moduation) and (inverse propogation)

Gs = \sum_e Me r_es $= \sum_e M_e r_{eo} = \sum_e m_e \otimes r_{es} = MR$ (modulation)

Gs = pinv(A) X $= A^{\ddagger} X = w^T X =$ WX (inverse propagation)

where, w is a [d x #S] spatial filter matrix for each source

The estimate in (modulation) depends on the assumed target, y, through Me|y

Thus, if we assume the true target gives the most similar activation we can select via:

Yest = \argmax_y sim(WX, M_y R)

where sim(a,b) is some measure of the similarity of it's inputs.
Using the inner product as this similarity measure we have:

Yest = \argmax_y w X M_y r

Note we can re-structure the computation of w X M_y r in different ways, depending on our computational needs, including:

w' X M r = Tr(w' X M r) = Tr(Cxy r w') = < Cxy, r w' >

where, Tr(.) is the trace operator, and
< a, b > is the frobenius norm (sum element-wise products)

Basically here, we first compute the ERP (Cxy) by convolving with the stimulus sequence My and then compute the similarity to the template ERP (r w').

It is computationally more efficient to reverse this order, i.e. first convolve the data with the termplate erp (r w') and then multiply by the stimulus sequence to compute the final similarity, i.e.

w' X M r = w' X (m*r) = w' (X*r) m = (w' (X*r)) m

where a*b represents the convolution of a and b

Finally we can write this as:
fe = w' (X*r) = X*(rw')

where fe is the convolution of the data with the template response.
This is an estimate of the similarity of the data to the template response for every sample, i.e. the predicted stimulus response

fy = fe m_y

where fy is the similarity of the predicted stimulus to the input stimulus for target y,
i.e. that is the predicted output

Nomenclature

d - number channels
tau - EEG response length in samples
#Tr - number of trials / epoch
#E - number of brain-stimulus event types (e.g. 2 = long+short)
#Y - number of targets to select from. (e.g. number of letters to select in speller)
#M - number of possible models (when train multiple models)
#S - number of distinct brain sources activited by the stimuli (when have multiple source regions, e.g. in tactile)

X = [d x tau x ... the Raw Channels By Time By Epochs Data

Me = [#E x #Y x ... The Brain Trigger Events Sequence...
#E Types Of Events x #Y possible Targets

g = [#S x T x #Y] Evoked brain activity over time for each source for each possible output target.
(i.e. the temporal response-templates)

gs - [1 x T x #Y] evoked activity for source s

Minimal info needed to fit the CCA model

summary statistics = {
Cxx = [d x d], spatial cross channel covariance
Cyy = [#E x tau x #E x tau x #M], cross event/time covariance (for each possible model)
Cxy = [d x tau x #E x #M] spatial-temporal (channel, stimulus) cross covariance (for each possible model)
}

We = [d x tau x #E x #M] The Trained (set Of #M) Models.
Each Model Is A Channels By Tau Samples By #E Brain Event Types Matrix

fe = [#E x #Tr] predicted stimulus event code, i.e. the stimulus score. Higher = more likely to be that stimulus event type.

fy = [#Y x #Tr] predicted output selection, i.e. the score for possible targets. Higher = more likely to be that target stimulus.

$\hat{y}$ = Yest = [#Y x ...] Selected output (binary encoded), alt is [1 x 1] as index

Perr = [1 x 1] Probability of error for the selected output

System / Algorithm Phases

1) Model-fitting (supervised)

INPUT: Stim_time, Event_type, Data (sliced)

X = [d x tau x #Tr]   Me = [#E x 1 x #Tr]

OUTPUT: decoder weight matrix

We = [d x tau x #E]

2) Model fitting (unsupervised) a.k.a. zero train

INPUT: Stim_time, Event_type (all possible targets), Data (sliced)

X = [d x tau x #Tr]   Me = [#E x #Y x #Tr]

OUTPUT: decoder weight matrix   We = [d x tau x #E]

Yest = [#Y x ...]   Selected output (binary encoded), alt is [1 x 1] as index

Perr = [1 x 1]   Probability of error for the selected output

3) Prediction / output selection

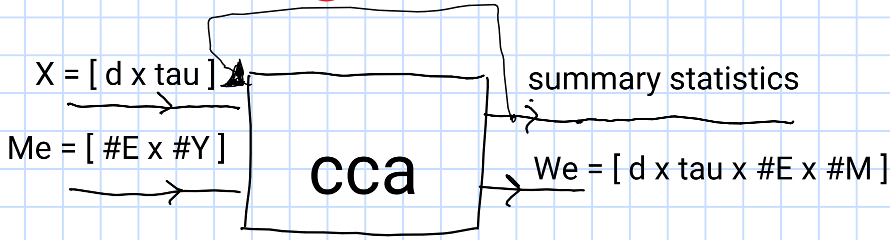
INPUT: Stim_time, Event_type (all possible targets), Data (sliced)

X = [d x tau x #Tr]   Me = [#E x #Y x #Tr]

OUTPUT: Yest = [#Y x ...]   Selected output (binary encoded), alt is [1 x 1] as index

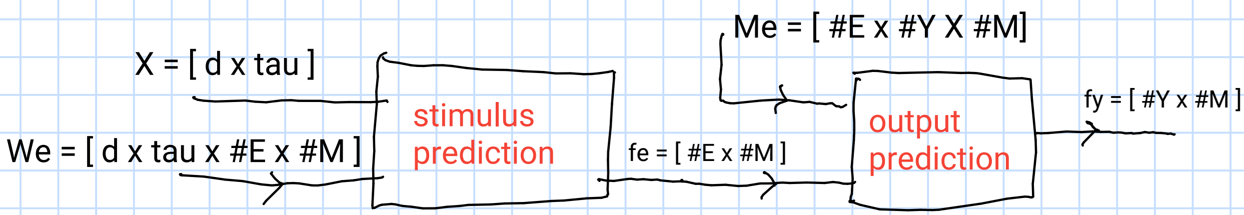
Perr = [1 x 1]   Probability of error for the selected output

Model Fitting

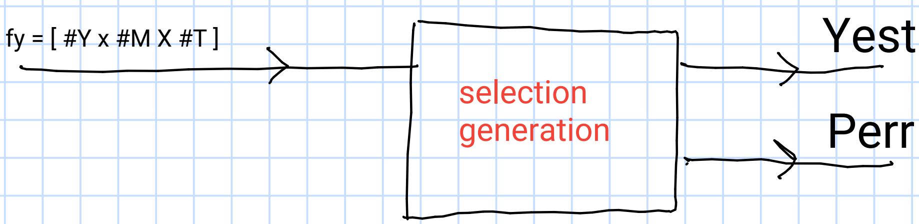


Prediction

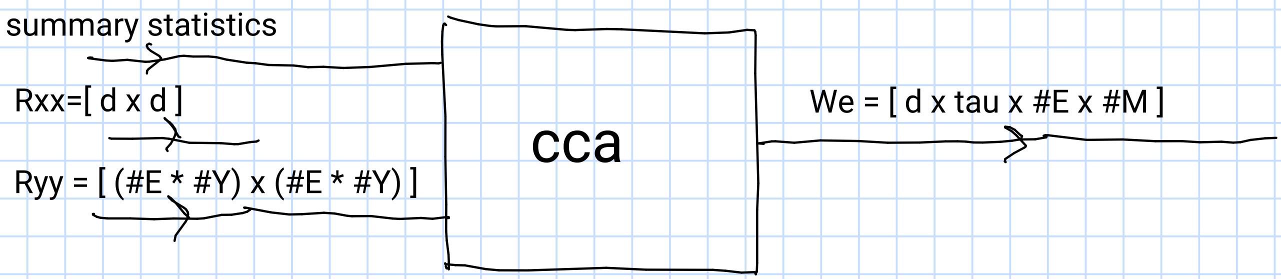
Target Score Computation



Output Selection

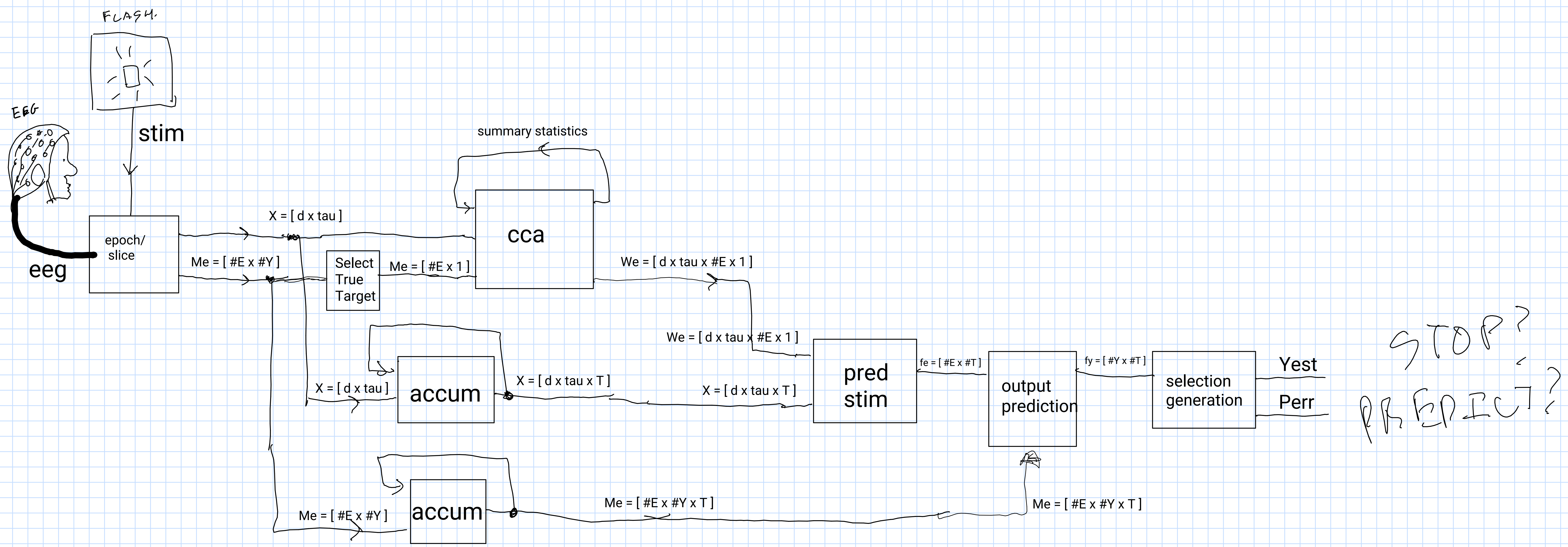


Adapatation

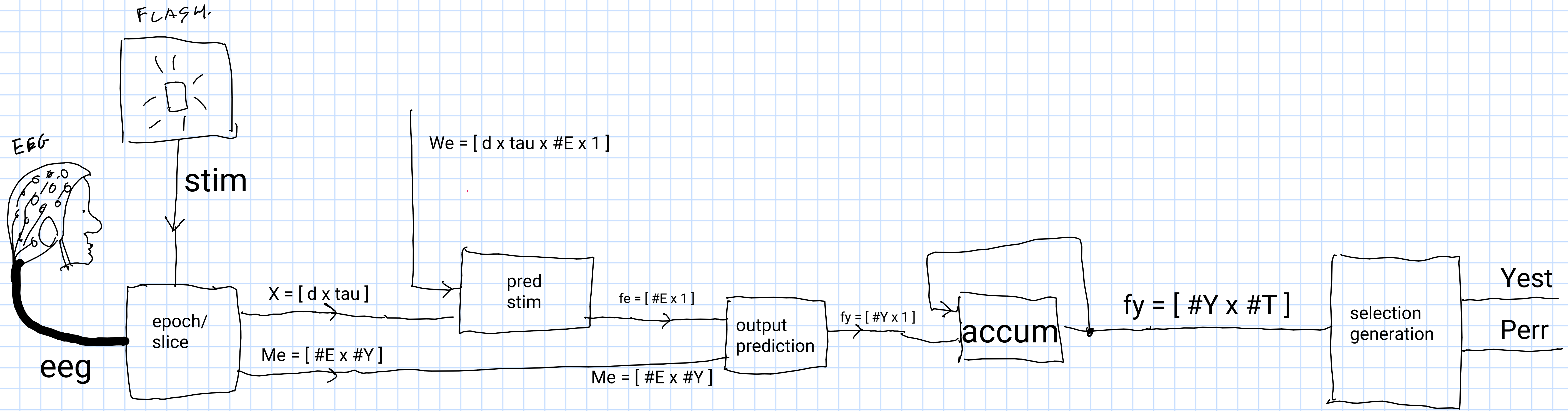


Architecture

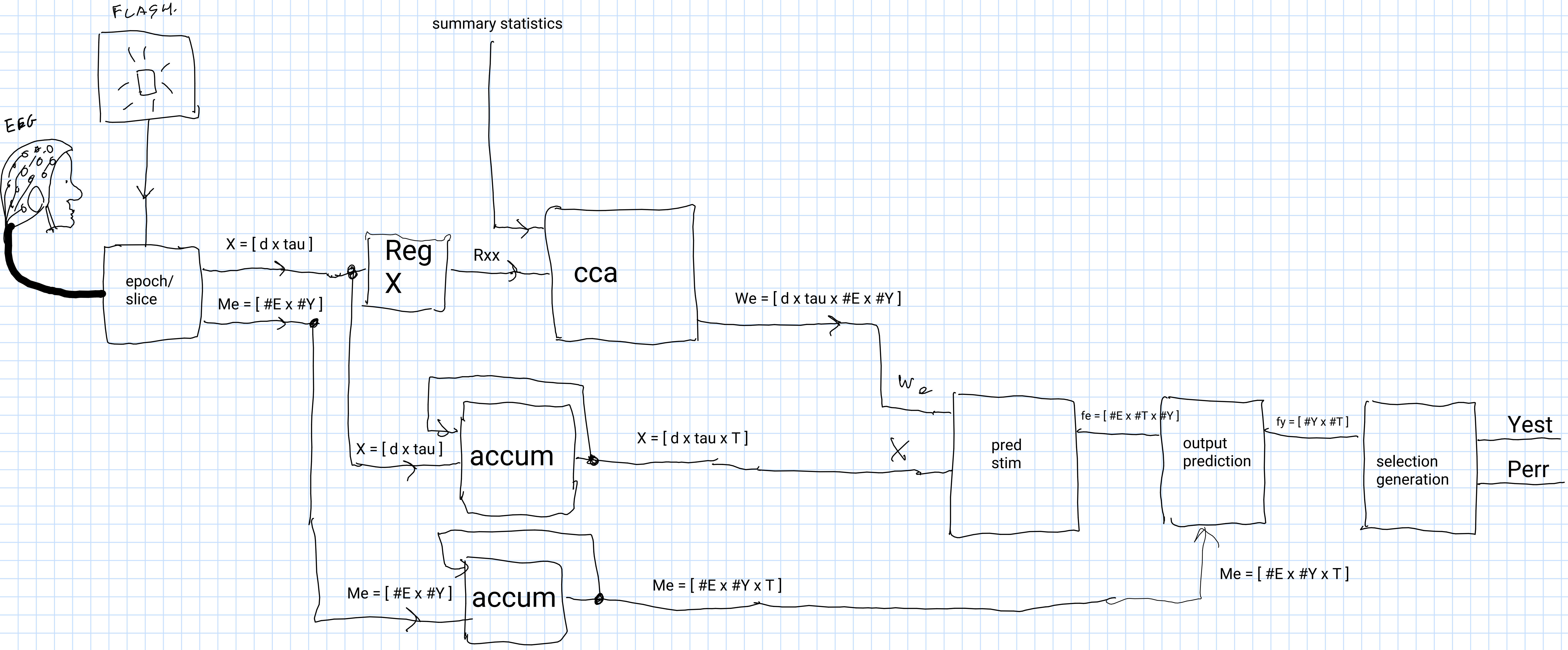
model fitting + output generation (supervised)



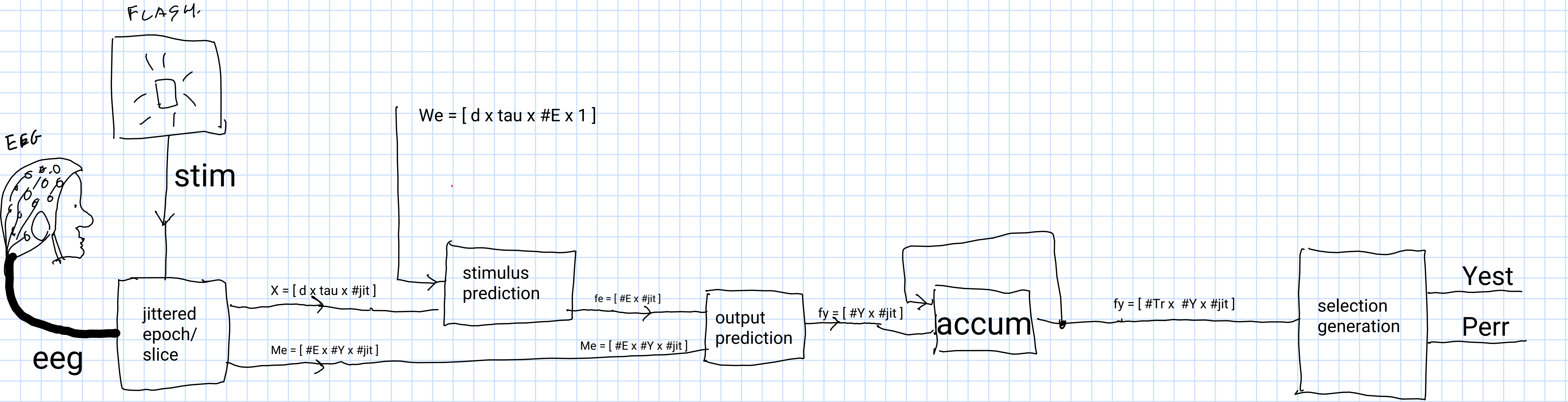
stimilus prediction + output generation



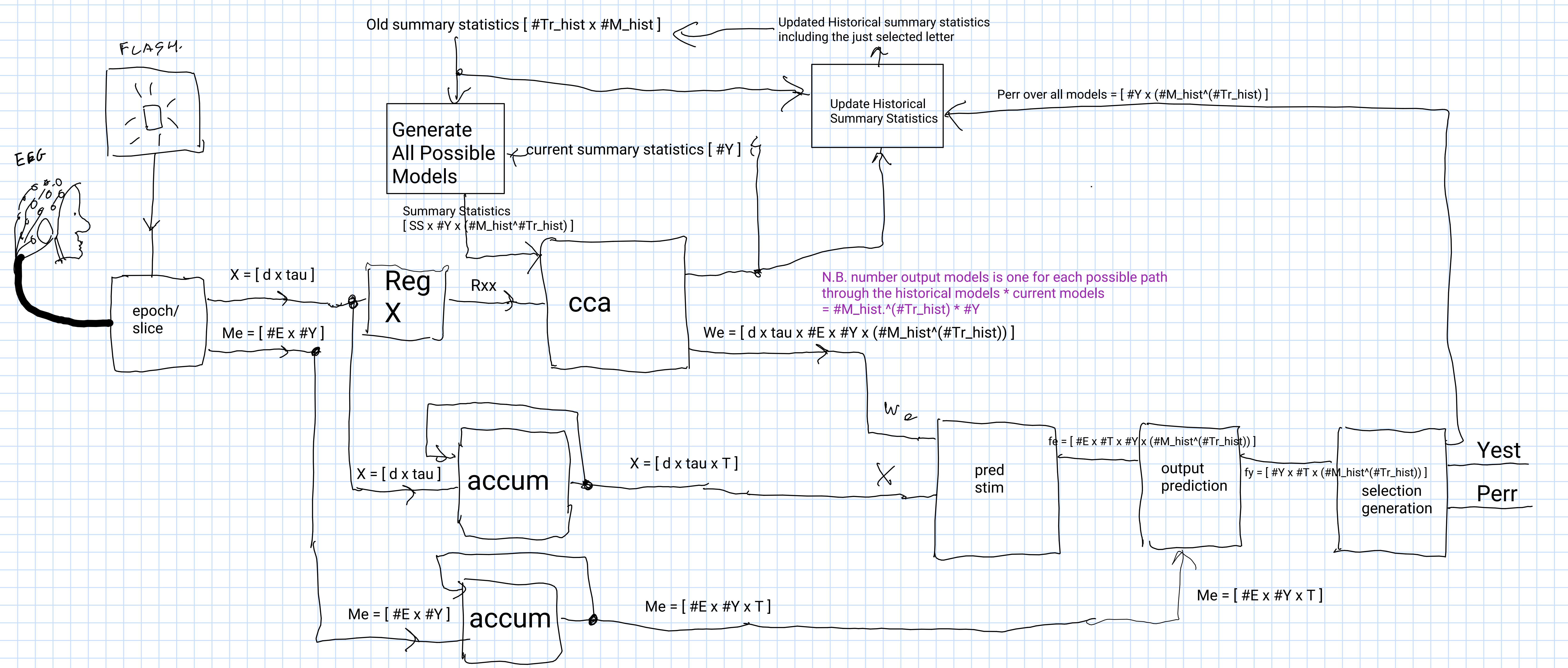
(unsupervised) model adaptation + output generation

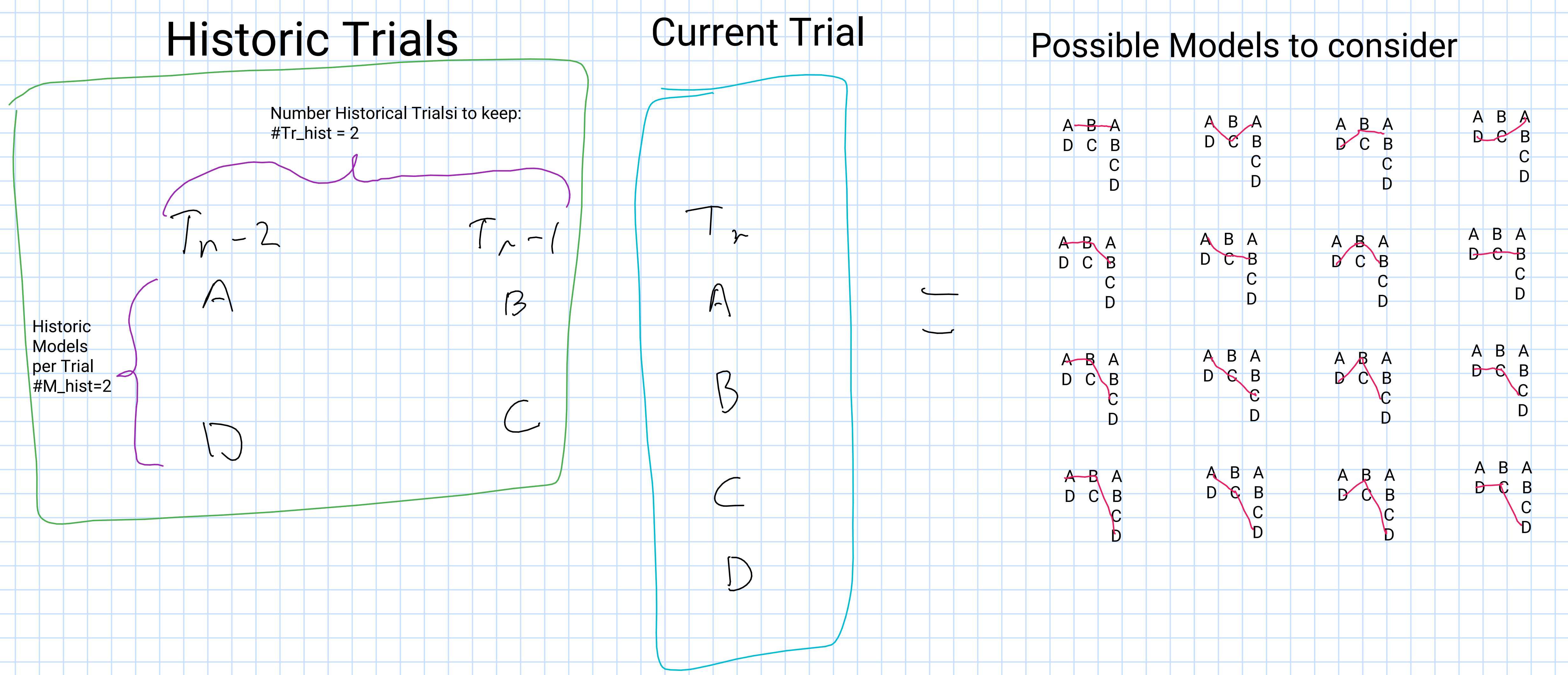


stimilus prediction + output generation. jittered/async



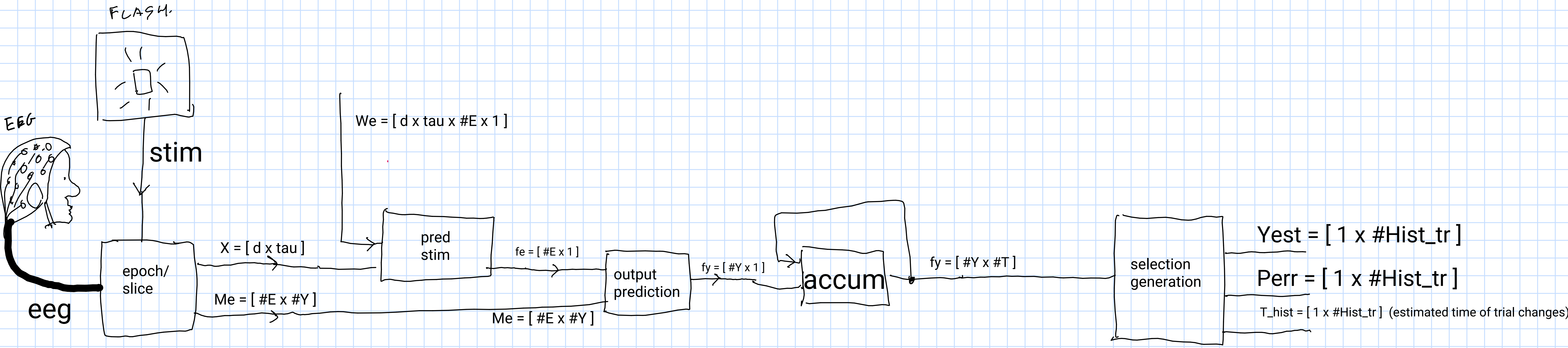
Non-Stop Learning (multi-trial zero-train)



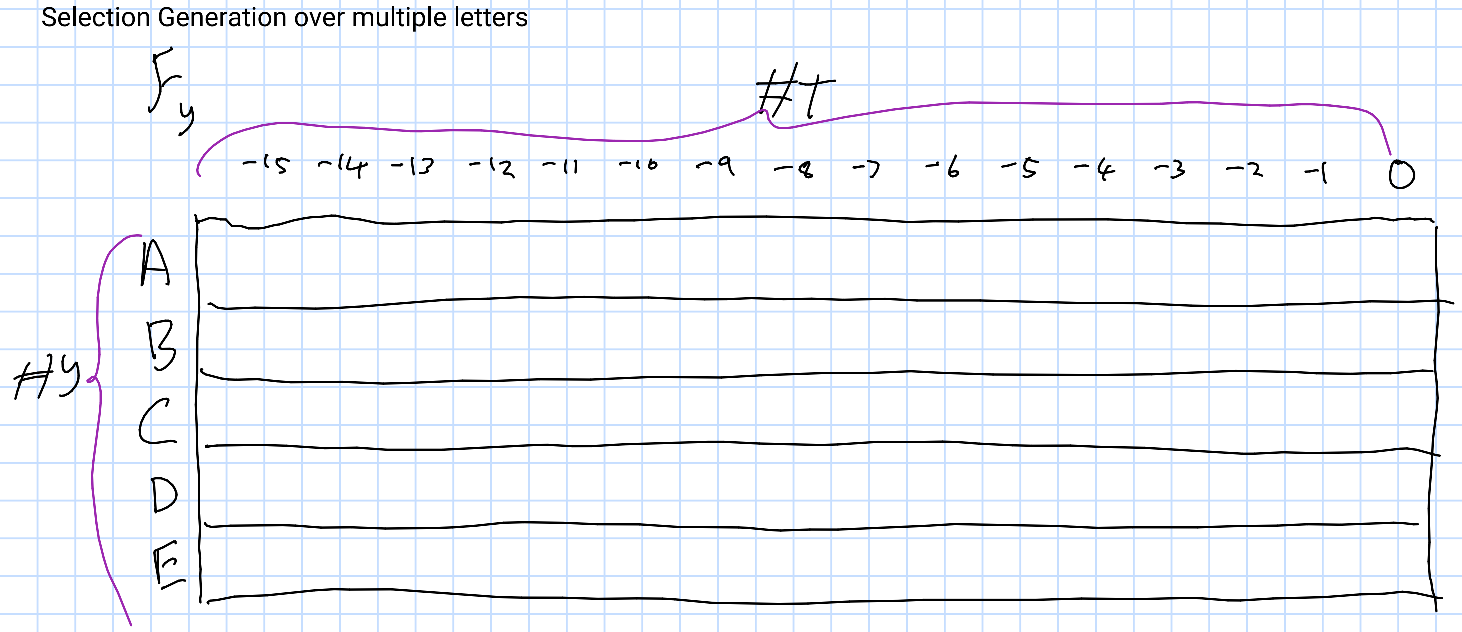


Example History Update Code: (Beam-search)
Add the #M_hist most probable letters over all possible models.
Perr(y) = \sum_M(Perr(M) s.t. M_curr==y)
where, M_curr is the output considered in this model at the current time
selected = top #M_hist of Perr

non-stop asynchronous decoding

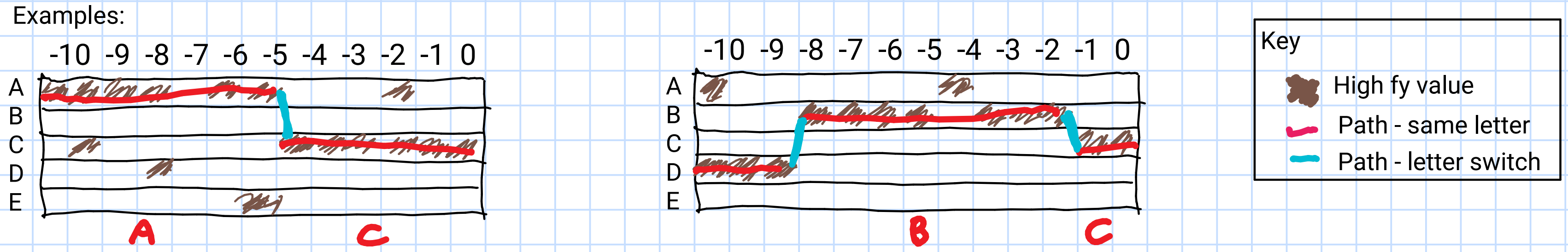


Selection Generation over multiple letters



Decoding - Find minimum cost horizontial path through the fy

Examples:



Opt Path:
Represent path, l, by [1 x #T] which indexs target letter at each time.
Then

l* = \argmax_l \sum_t fy(l(t)) + R*(l(t)~=l(t-1))

$$= \underset{l}{\mathrm{argmax}} \sum_t \underbrace{f_y(l(t))}_{\text{Reward for this letter at this time}} + \underbrace{R\left[1-\delta(l(t)-l(t-1))\right]}_{\text{Additional Reward for *not* changing letter}}$$

Efficient solution with dynamic-programming. (O(#Y) / time-point)