Machine Learning at ALS — First Studies

Simon C. Leemann

ALS Accelerator Physics, ATAP Division, Lawrence Berkeley National Laboratory

February 6, 2019

Contributions from: Hiroshi Nishimura, Shuai Liu, David Shapiro, Changchun Sun

BERKELEY LAB

U.S. DEPARTMENT OF ENERGY | Office of Science

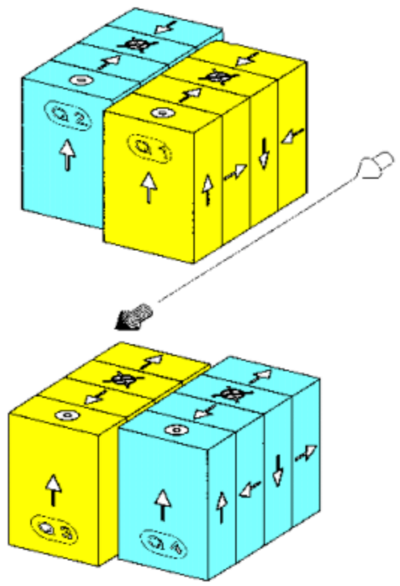
ALS ADVANCED LIGHT SOURCE

Background: ML @ ALS

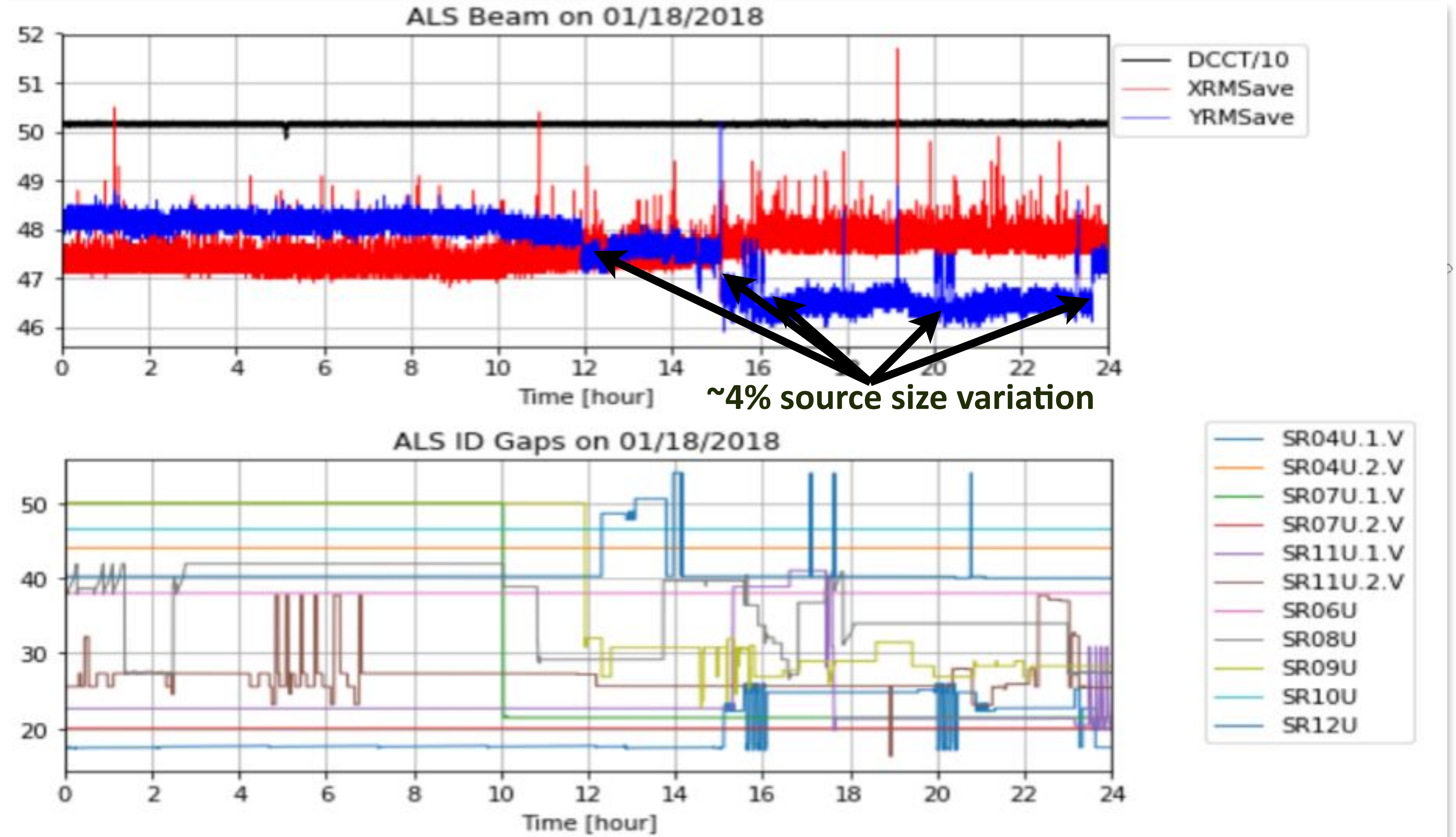
- ALS–SSRL collaboration on *"Beam Based Optimization and Machine Learning for Synchrotrons"*
  - SSRL: X. Huang, J. Safranek
  - ALS: S.C. Leemann, A. Hexemer
- Recently received DOE BES/ASCR funding
  - $660k for ALS work package over 2 years

A first problem for ML @ ALS: beam size vs. IDs

- ALS has various IDs with constantly changing gaps and phases
  - There are feed-forwards & feedbacks stabilizing the source positions/angles
  - There are feed-forwards & feedbacks stabilizing optics at the source points → local (beta) and global (tune)
  - There are feed-forwards to stabilize the source size
    - these require recording lookup tables
    - tables are imperfect and machine drifts over time
- Nevertheless, during routine user ops source size changes @ BL3.1

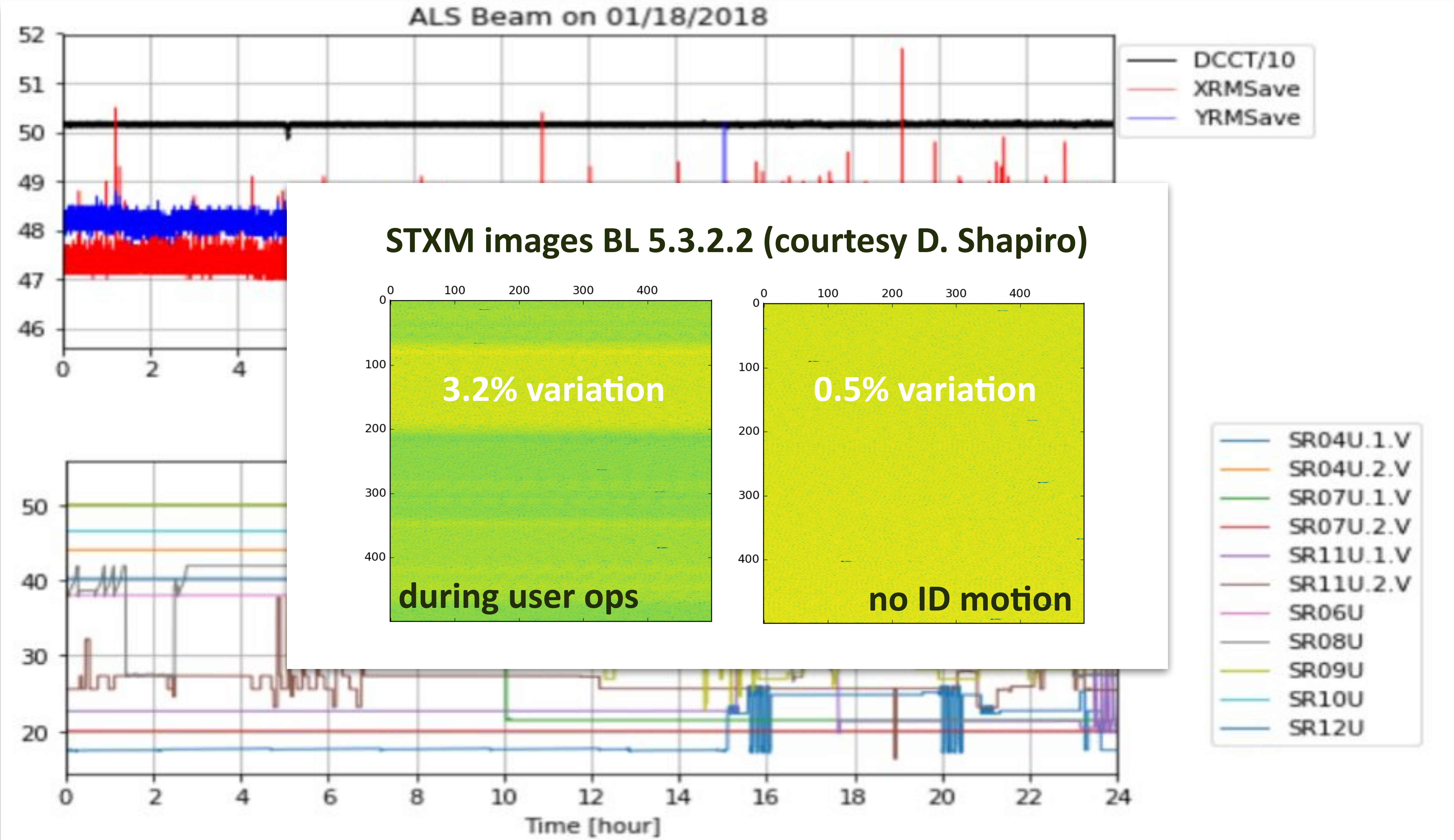


A first problem for ML @ ALS: beam size vs. IDs



Courtesy: H. Nishimura

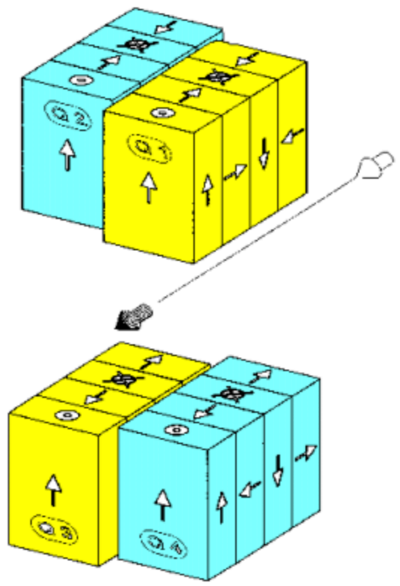
A first problem for ML @ ALS: beam size vs. IDs



Courtesy: H. Nishimura

A first problem for ML @ ALS: beam size vs. IDs

- ALS has various IDs with constantly changing gaps and phases
  - There are feed-forwards & feedbacks stabilizing the source positions/angles
  - There are feed-forwards & feedbacks stabilizing optics at the source points → local (beta) and global (tune)
  - There are feed-forwards to stabilize the source size
    - these require recording lookup tables
    - tables are imperfect and machine drifts over time

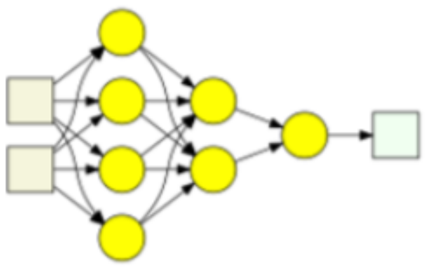


- Nevertheless, during routine user ops source size changes @ BL3.1
- ML can be used to predict beam size change as a function of arbitrary ID configuration → adjust skew quadrupole configuration slightly to compensate for ID-induced source size changes

Why rely on ML?

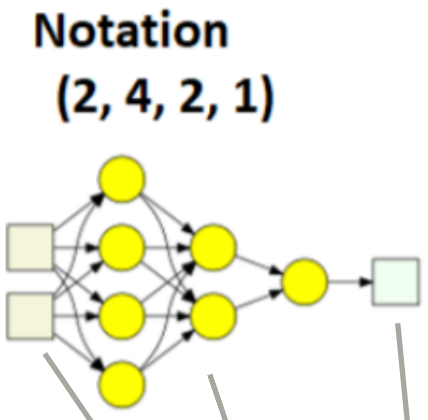
- ML does not require understanding underlying physics
- ML can model highly nonlinear processes and is extremely flexible
  - vary number of hidden layers & nodes
  - "hyper parameter tuning" → no general guiding principle, but gain lots of intuition with growing experience

Notation
(2, 4, 2, 1)



Why rely on ML?

- ML does not require understanding underlying physics
- ML can model highly nonlinear processes and is extremely flexible
  - vary number of hidden layers & nodes
  - "hyper parameter tuning" → no general guiding principle, but gain lots of intuition with growing experience
  - MSE indicates "fit quality" and reveals over-fitting

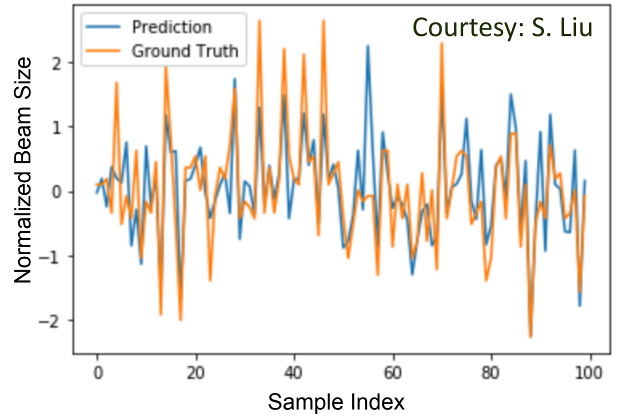


$$\mathrm{MSE} = \min_{w \in \mathcal{R}^m} \left[\sum_{i=1}^{n} (f(x_i, w) - y_i)^2 \right]$$

Why rely on ML?

- ML does not require understanding underlying physics
- ML can model highly nonlinear processes and is extremely flexible
  - vary number of hidden layers & nodes
  - "hyper parameter tuning" → no general guiding principle, but gain lots of intuition with growing experience
  - MSE indicates "fit quality" and reveals over-fitting
- But isn't this the same as fitting arbitrary-order polynomials?

$$\mathrm{MSE} = \min_{w \in \mathcal{R}^{ml}} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{m} \sum_{k=1}^{l} w_{jk} x_j^k + b)^2 \right]$$



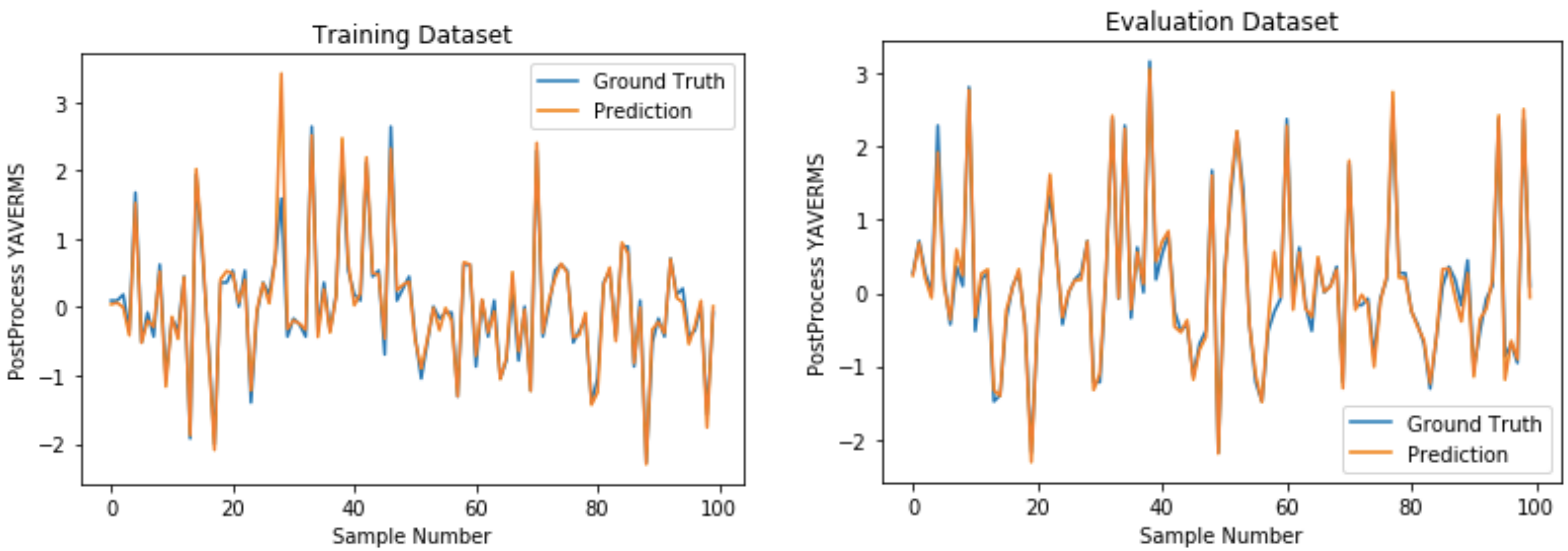
MSE: 0.353

Polynomial Order	Training MSE	Evaluation MSE
1	0.476	0.478
2	0.428	0.429
3	0.381	0.383
4	0.367	0.368
5	0.357	0.359
6	0.353	0.354
Deep Learning	0.0230	0.0232

No!

A first ML study on this problem: prediction

- ML requires reproducible events and ideally needs large data sets for training ➔ ALS has huge amounts of data to offer
  - 26 ID parameters ("input") & 2 beam sizes @ BL3.1 ("output")
  - Recorded 8 Msamples @ 10 Hz ➔ 6 Msamples used for training, 2 Msamples for validation ➔ training took 30 min on powerful GPU, MSE is 0.02



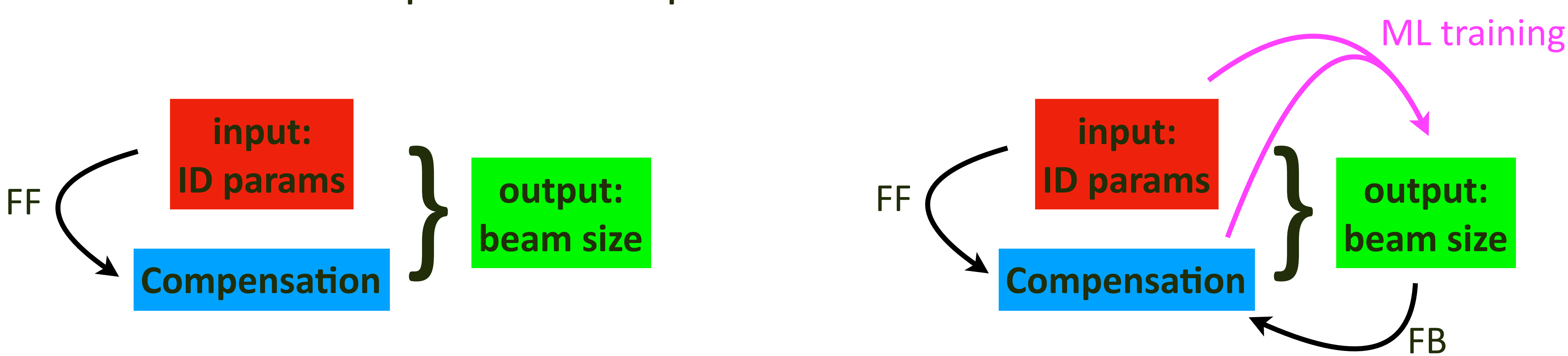
MSE: 0.0230

MSE: 0.0232

Courtesy: S. Liu

A first ML study on this problem: prediction

- ML requires reproducible events and ideally needs large data sets for training → ALS has huge amounts of data to offer
  - 26 ID parameters ("input") & 2 beam sizes @ BL3.1 ("output")
  - Recorded 8 Msamples @ 10 Hz → 6 Msamples used for training, 2 Msamples for validation → training took 30 min on powerful GPU, MSE is 0.02
- Now have lookup table → implement ML-based feed-forward



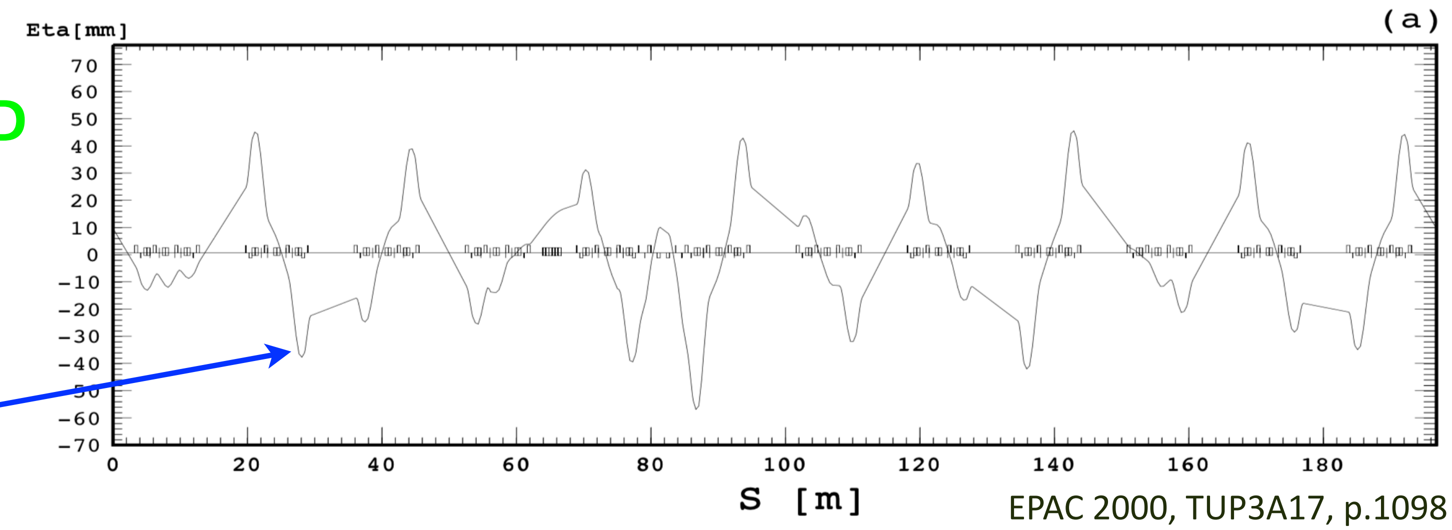
From prediction to correction

- Introduced a scaling ("DWP") to standard ALS dispersion wave (skew quadrupole excitation pattern) → allows adjusting vertical emittance evenly among all source points

SQSF SQSD

$$\vec{K} = \vec{K}_0 + (\chi_0 + \chi)\Delta\vec{K}, \quad \vec{K} \in \mathcal{R}^{16+16}$$

LOCO & Setup **DWP** **Dispersion Wave**



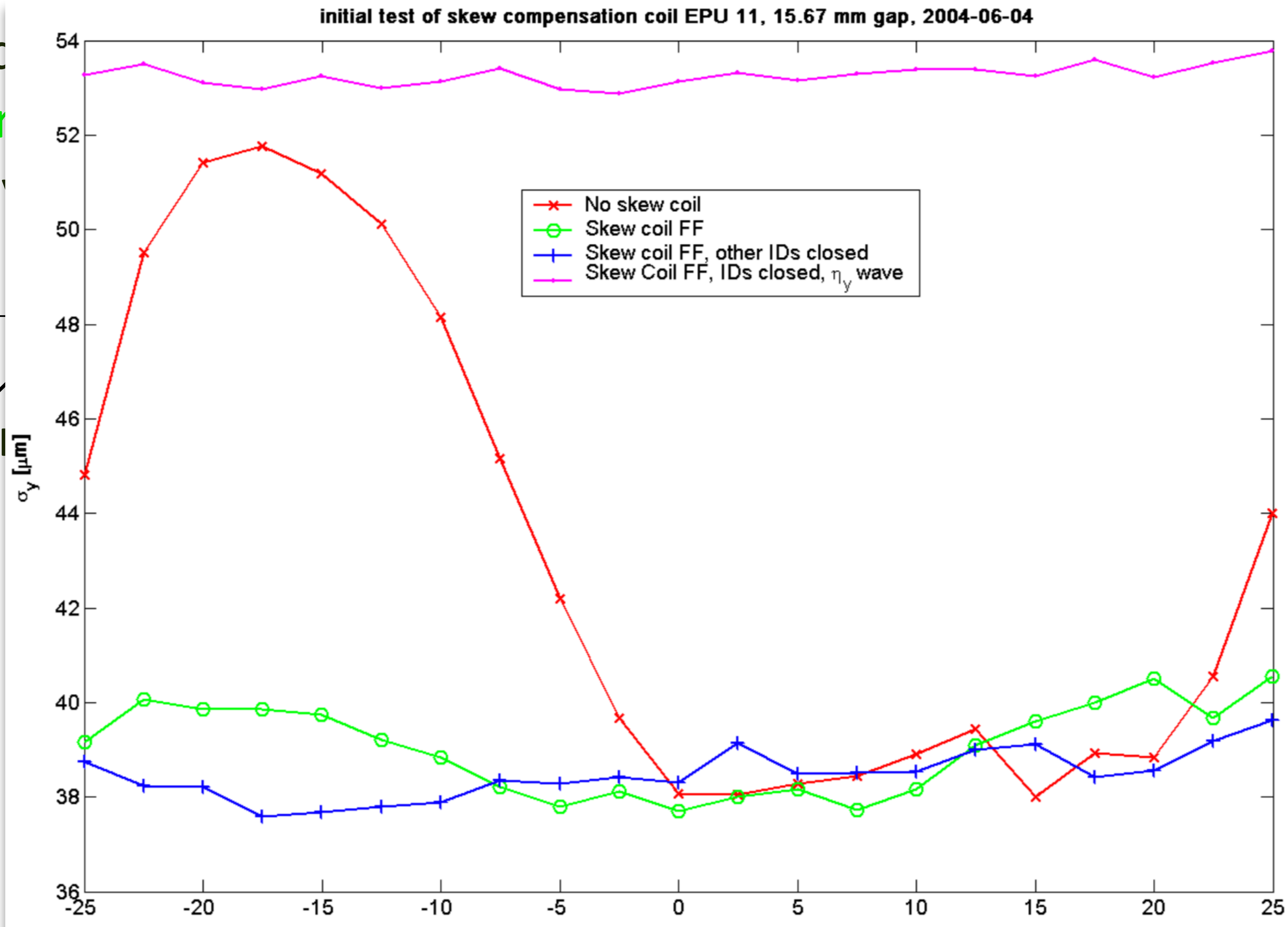
EPAC 2000, TUP3A17, p.1098

From prediction to correction

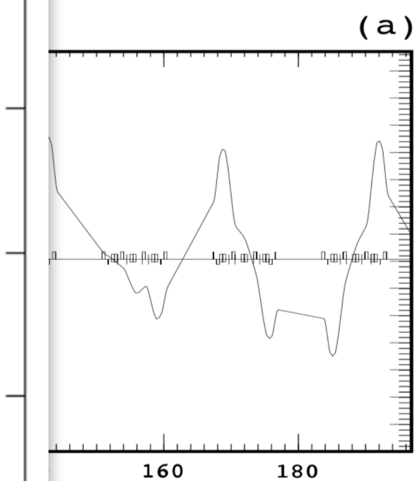
- Introd (skew quadr tance evenl

$\vec{K} = \vec{K}_0 +$

LOCO & Setu



EPAC 2004, MOPKF071, p.479



2000, TUP3A17, p.1098

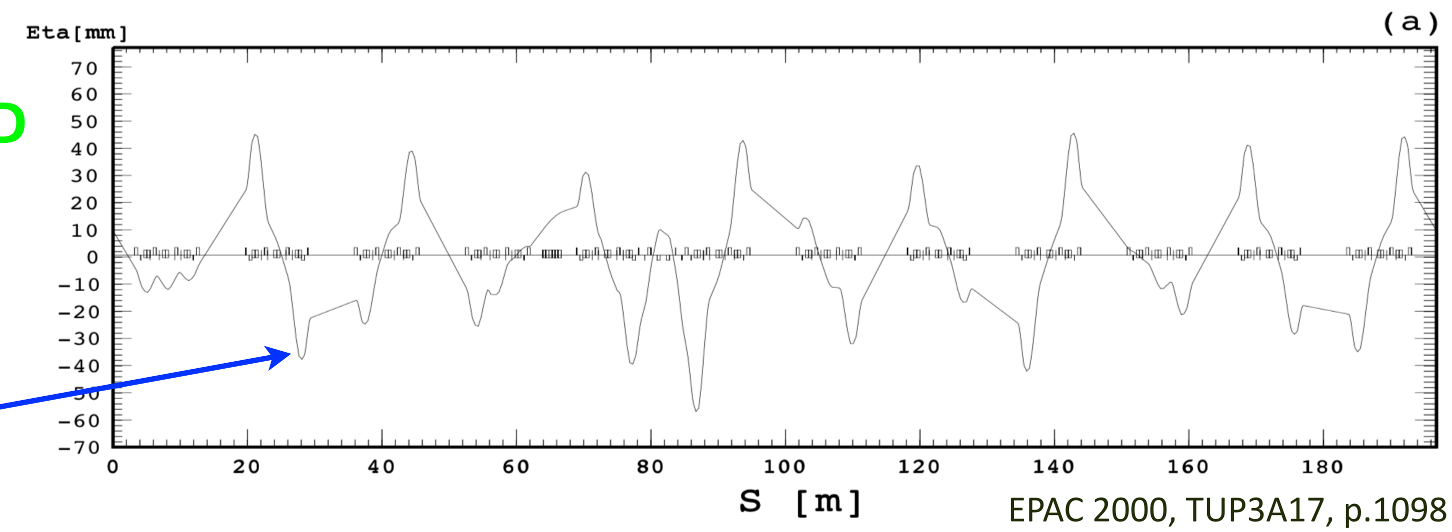
From prediction to correction

- Introduced a scaling ("DWP") to standard ALS dispersion wave (skew quadrupole excitation pattern) → allows adjusting vertical emittance evenly among all source points

SQSF SQSD

$$\vec{K} = \vec{K}_0 + (\chi_0 + \chi)\Delta\vec{K}, \quad \vec{K} \in \mathcal{R}^{16+16}$$

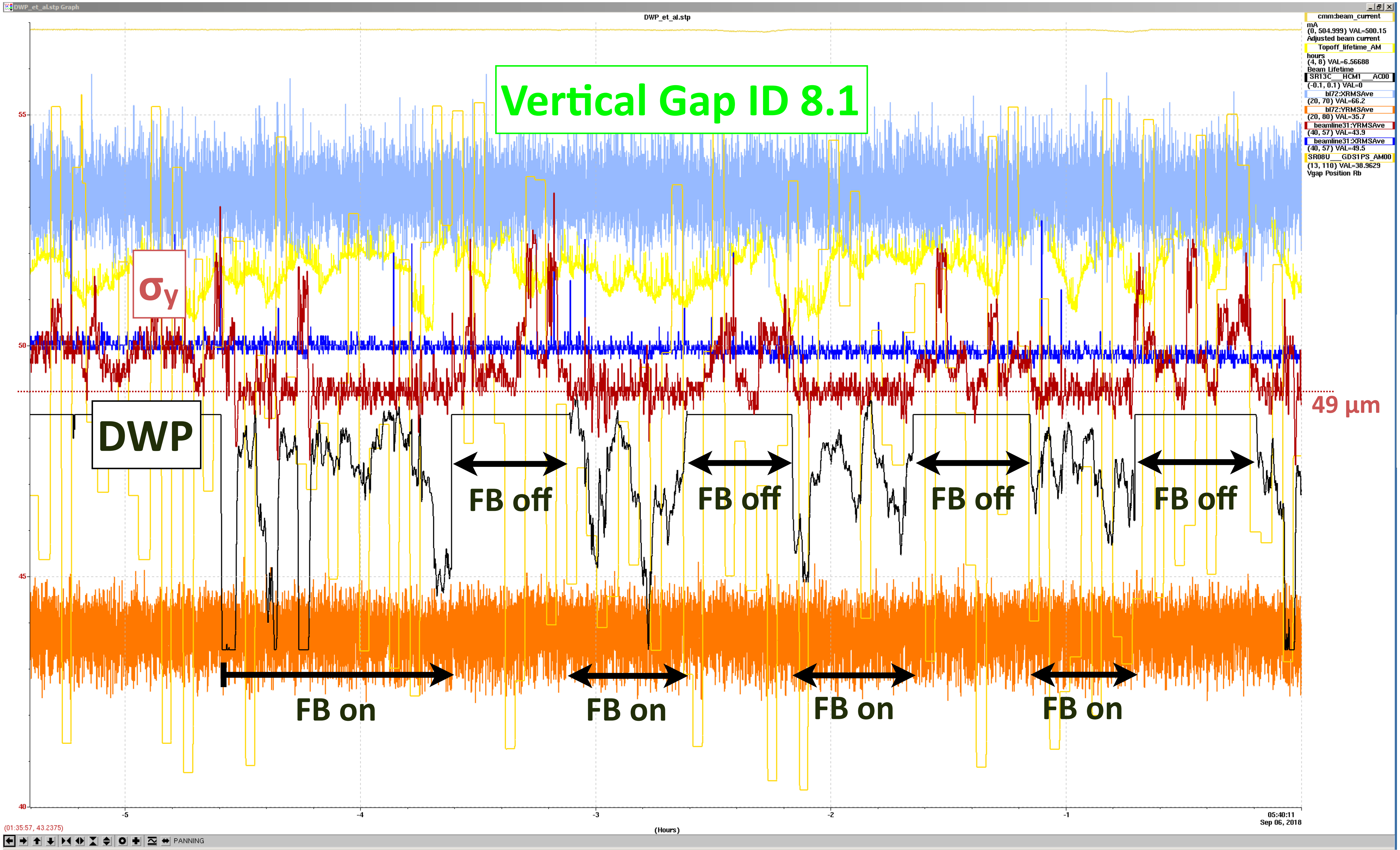
LOCO & Setup DWP Dispersion Wave



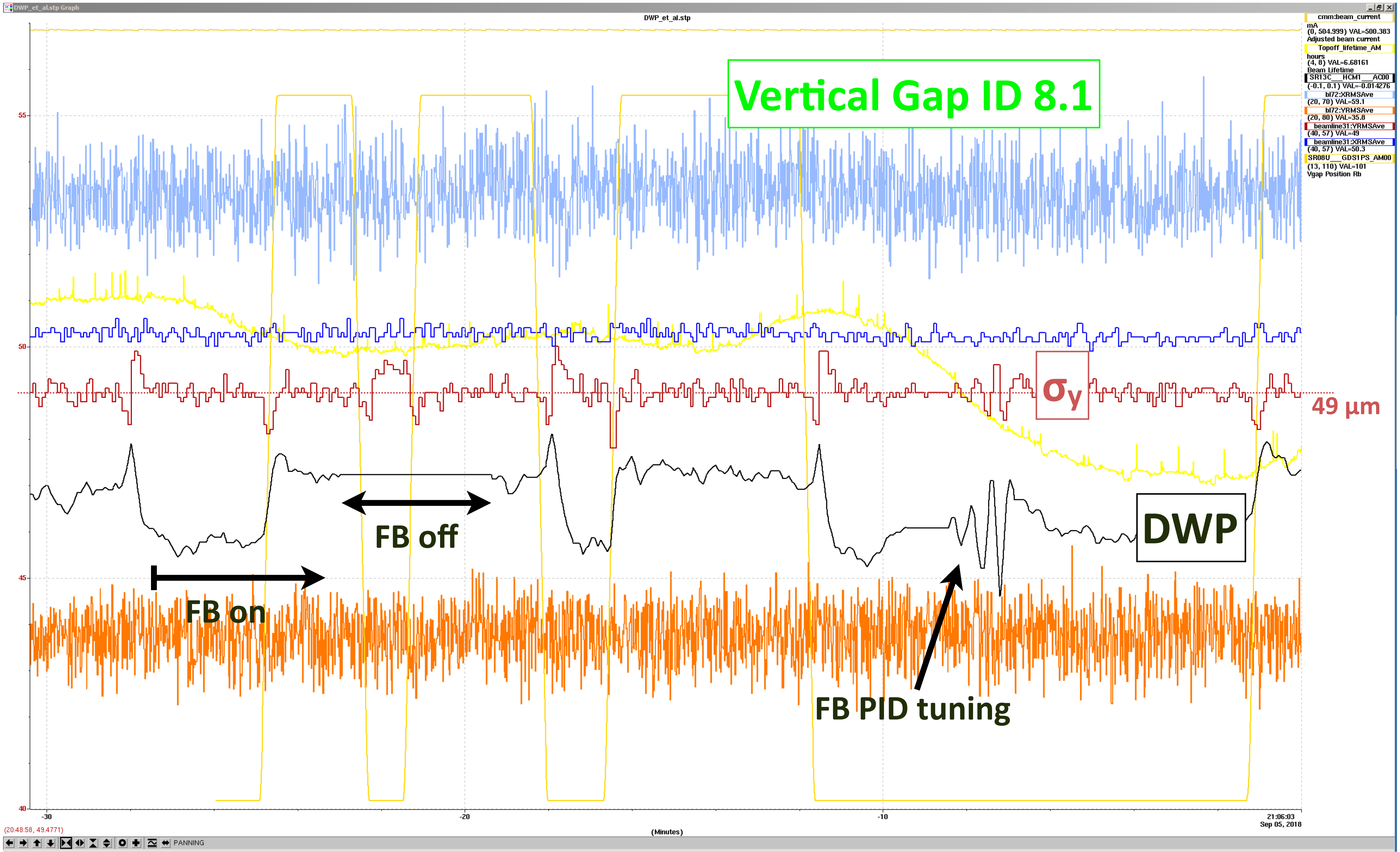
EPAC 2000, TUP3A17, p.1098

- Tested simple PID feedback loop designed to stabilize source size as measured at BL3.1 by adjusting DWP
- Run this feedback loop while scanning various ID configurations → acquire data at 10 Hz → input for training of NN (deep learning = DL)
- Result of DL is prediction for DWP required to keep beam size constant for arbitrary ID configurations

From prediction to correction

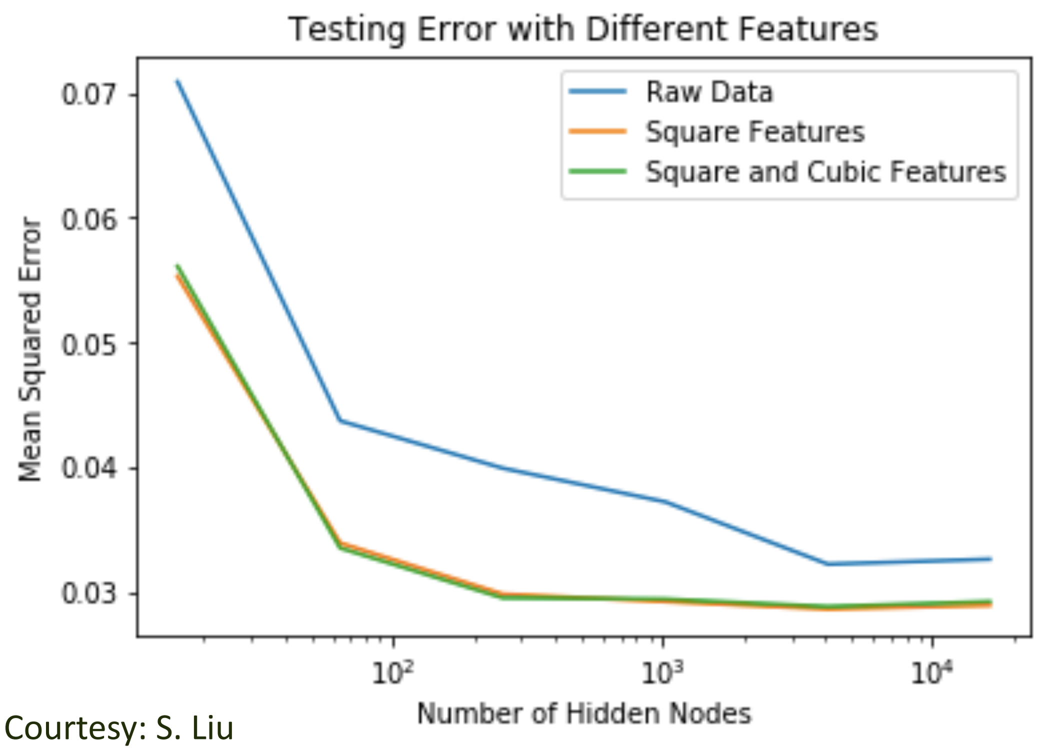


From prediction to correction



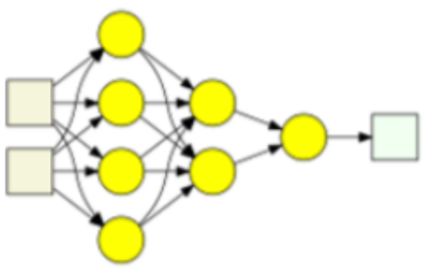
Learned many ML tricks with this first simple NN

- Increase nodes and layers until observe MSE convergence
- Add square/cubic features to inputs to increase convergence



Courtesy: S. Liu

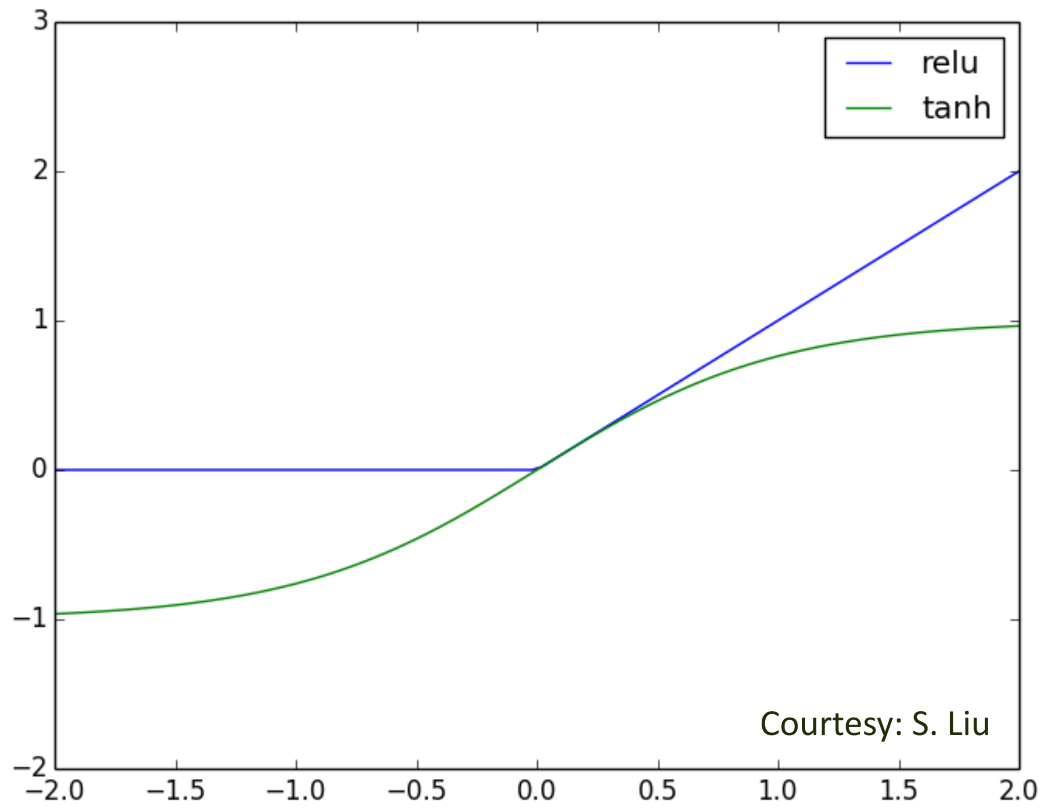
Notation
(2, 4, 2, 1)



	Raw Data		With Square Features	
Architecture	Training MSE	Evaluation MSE	Training MSE	Evaluation MSE
128-64	0.0265	0.0268	0.0257	0.0260
256-64	0.0243	0.0245	0.0259	0.0262
512-128	0.0243	0.0247	0.0243	0.0247
128-64-32	0.0238	0.0242	0.0243	0.0245
256-128-64	0.0236	0.0240	0.0240	0.0246
256-128-64-32	0.0245	0.0249	0.0245	0.0248

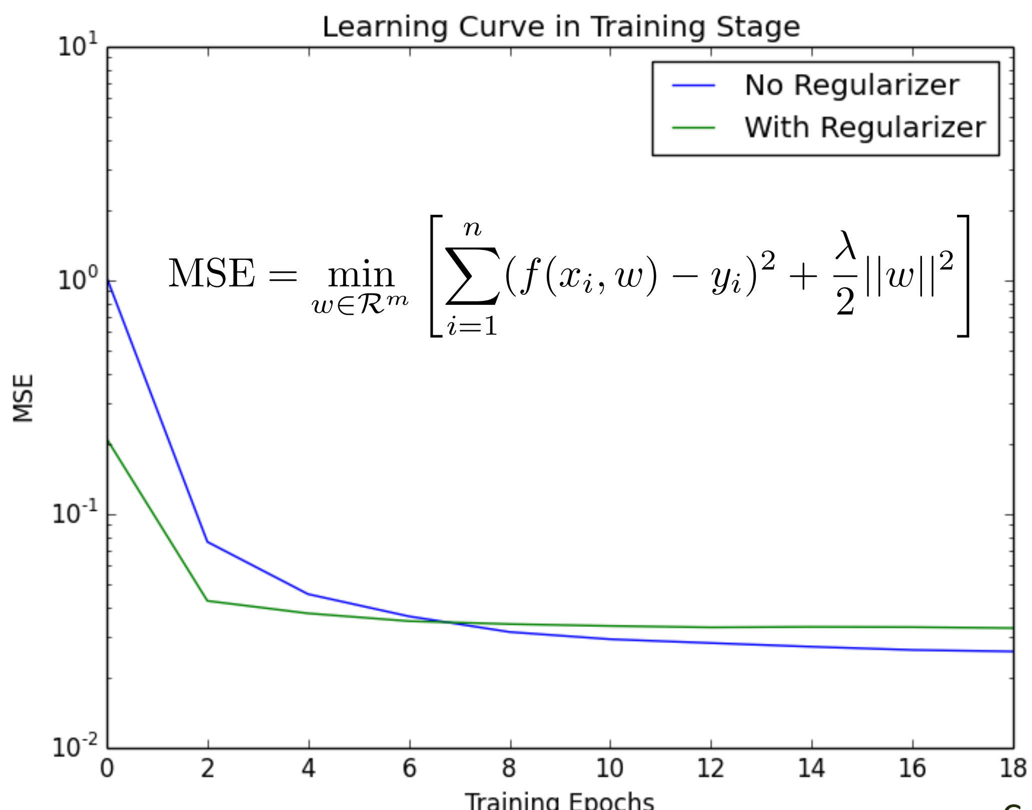
Learned many ML tricks with this first simple NN

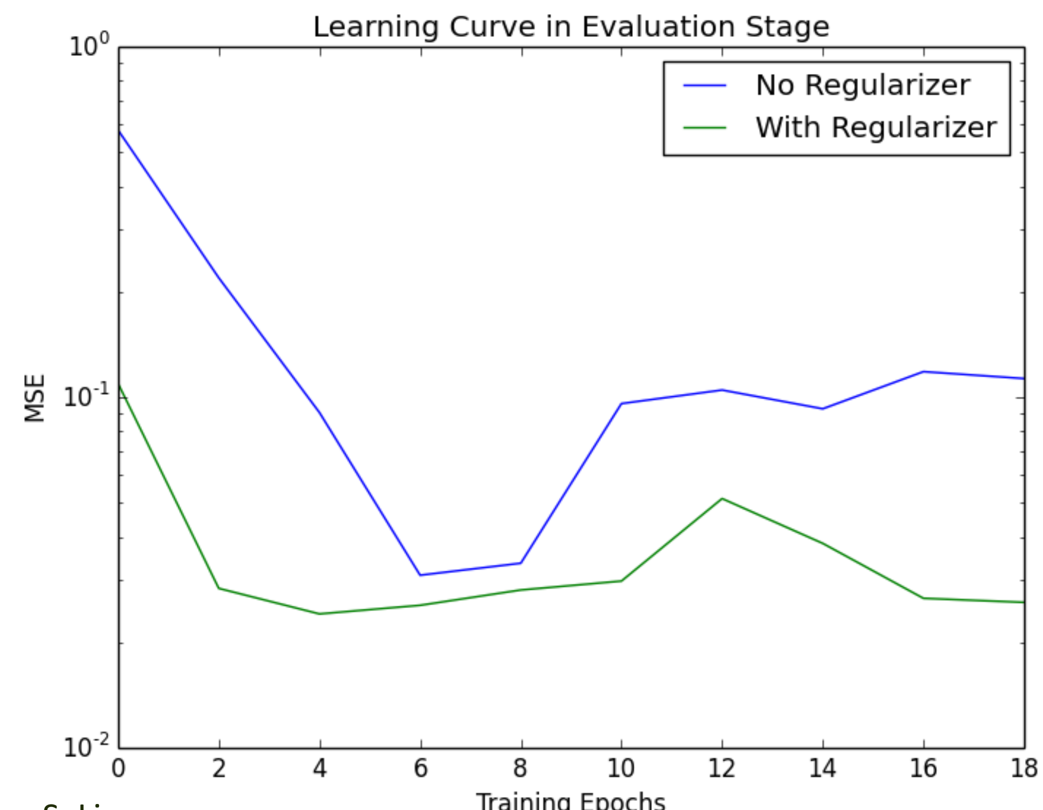
- Increase nodes and layers until observe MSE convergence
- Add square/cubic features to inputs to increase convergence
- Vary activation functions until best convergence achieved



Learned many ML tricks with this first simple NN

- Increase nodes and layers until observe MSE convergence
- Add square/cubic features to inputs to increase convergence
- Vary activation functions until best convergence achieved
- Regularization → avoid overfitting, increase robustness

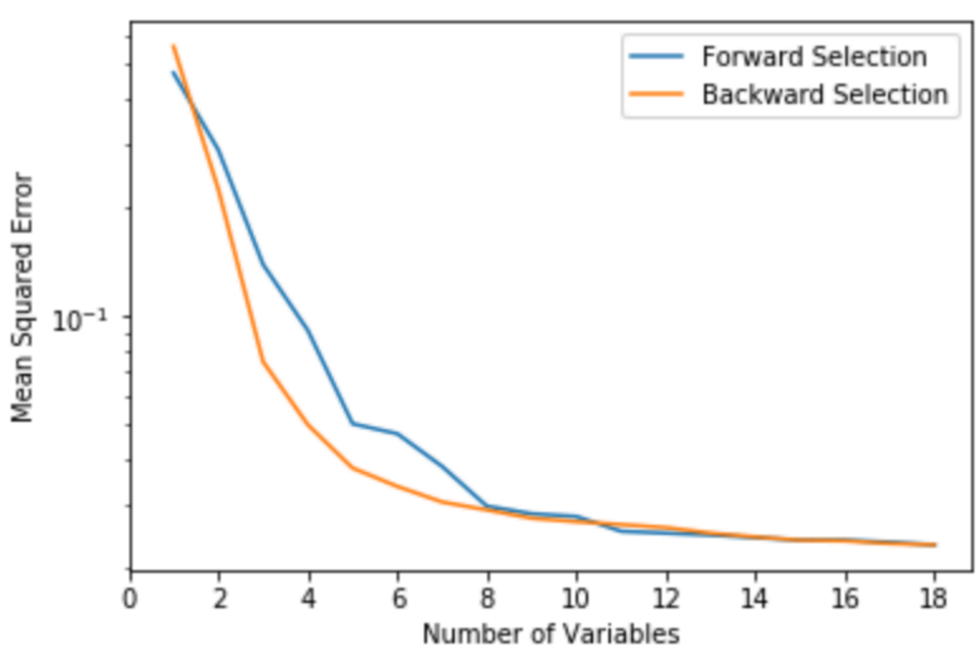




Courtesy: S. Liu

Learned many ML tricks with this first simple NN

- Increase nodes and layers until observe MSE convergence
- Add square/cubic features to inputs to increase convergence
- Vary activation functions until best convergence achieved
- Regularization ➔ avoid overfitting, increase robustness
- Forward/backward selection ➔ allows selecting relevant parameters



Courtesy: S. Liu

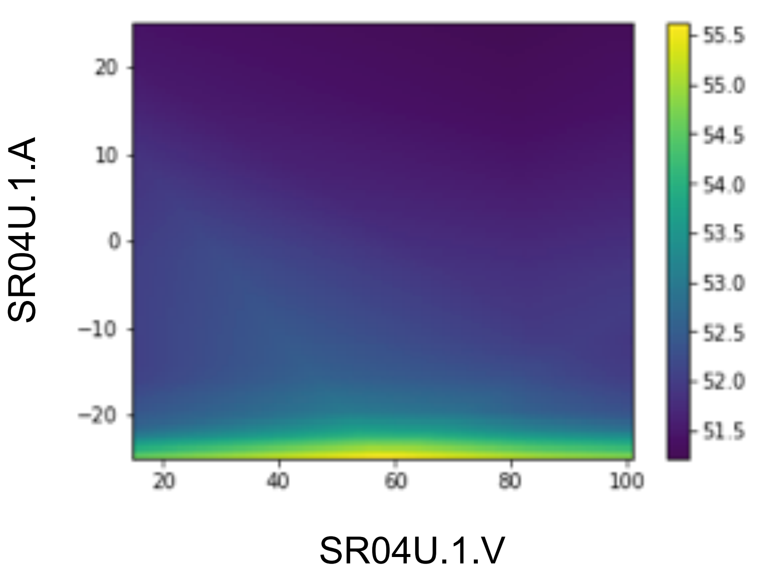
More Important

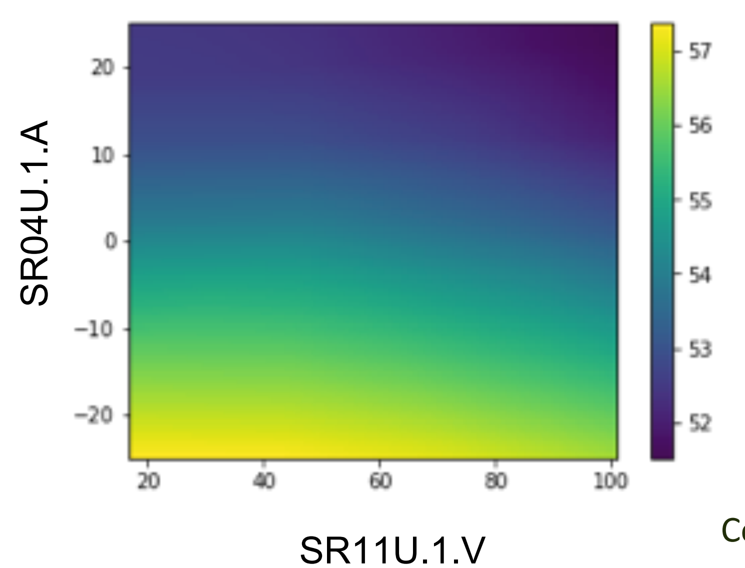
Forward Selection	Backward Selection
SR07U.1.A	SR04U.2.A
SR011U.2.A	SR011U.2.A
SR04U.2.A	SR12U
SR06U	SR06U
SR07U.1.V	SR07U.1.V
SR07U.2.A	SR10U
SR10U	SR07U.1.A
SR12U	SR07U.2.A
SR011U.1.A	SR011U.2.V
SR011U.2.V	SR011U.1.V
SR04U.2.V	SR04U.1.A
DCCT	SR08U
SR04U.1.A	SR011U.1.A
SR04U.1.V	DCCT
SR09U	SR04U.2.V
SR07U.2.V	SR09U
SR08U	SR07U.2.V
SR011U.1.V	SR011U.2.A

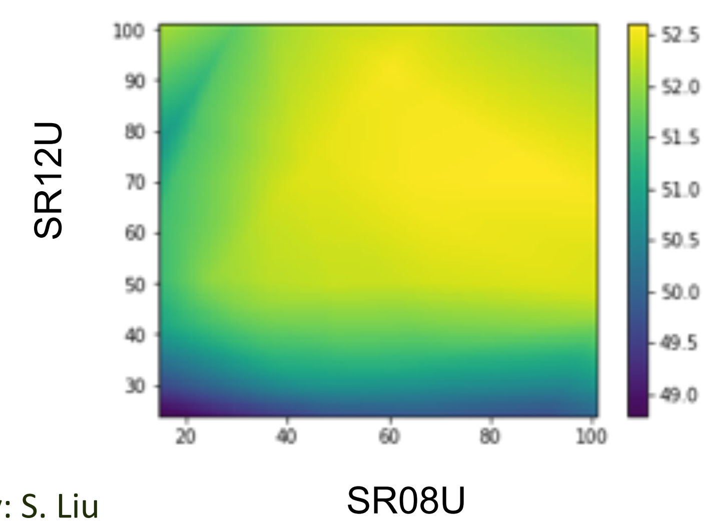
More Important

Learned many ML tricks with this first simple NN

- Increase nodes and layers until observe MSE convergence
- Add square/cubic features to inputs to increase convergence
- Vary activation functions until best convergence achieved
- Regularization → avoid overfitting, increase robustness
- Forward/backward selection → allows selecting relevant parameters
- Study interactions → parameter subspace for training?



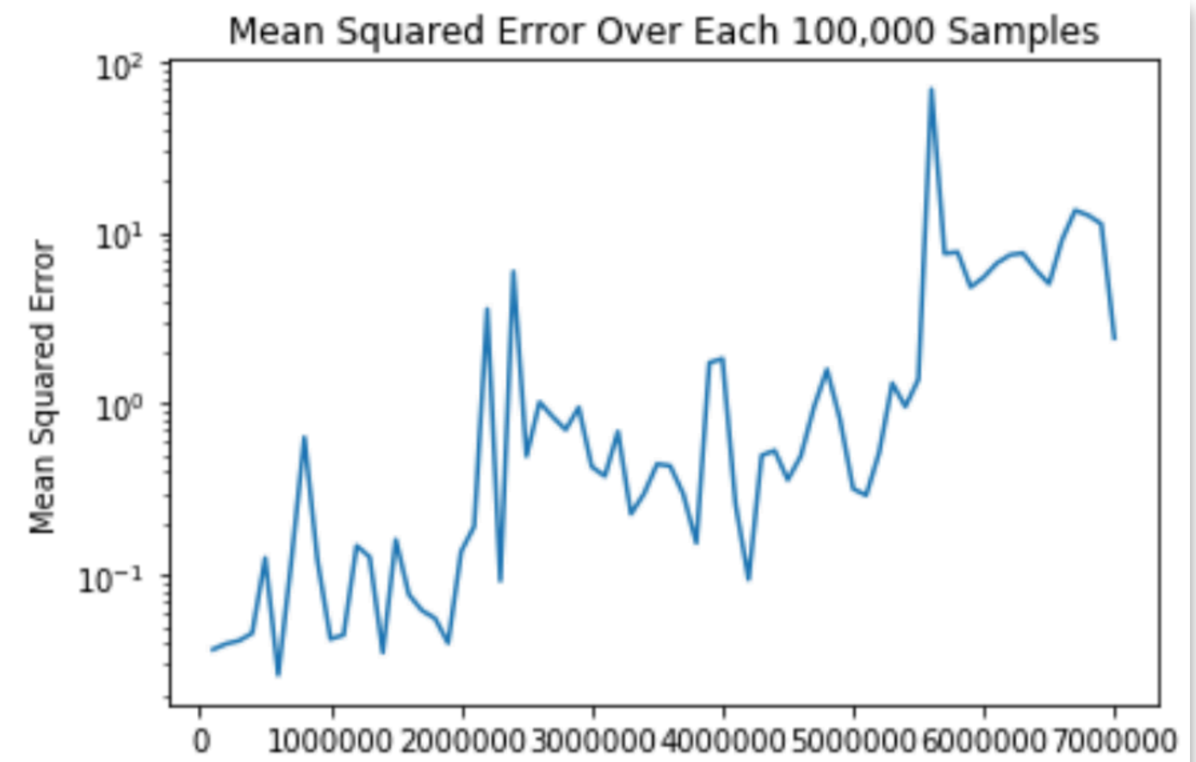




Courtesy: S. Liu

Learned many ML tricks with this first simple NN

- Increase nodes and layers until observe MSE convergence
- Add square/cubic features to inputs to increase convergence
- Vary activation functions until best convergence achieved
- Regularization → avoid overfitting, increase robustness
- Forward/backward selection → allow
- Study interactions → parameter subs
- Online learning → overcome drift

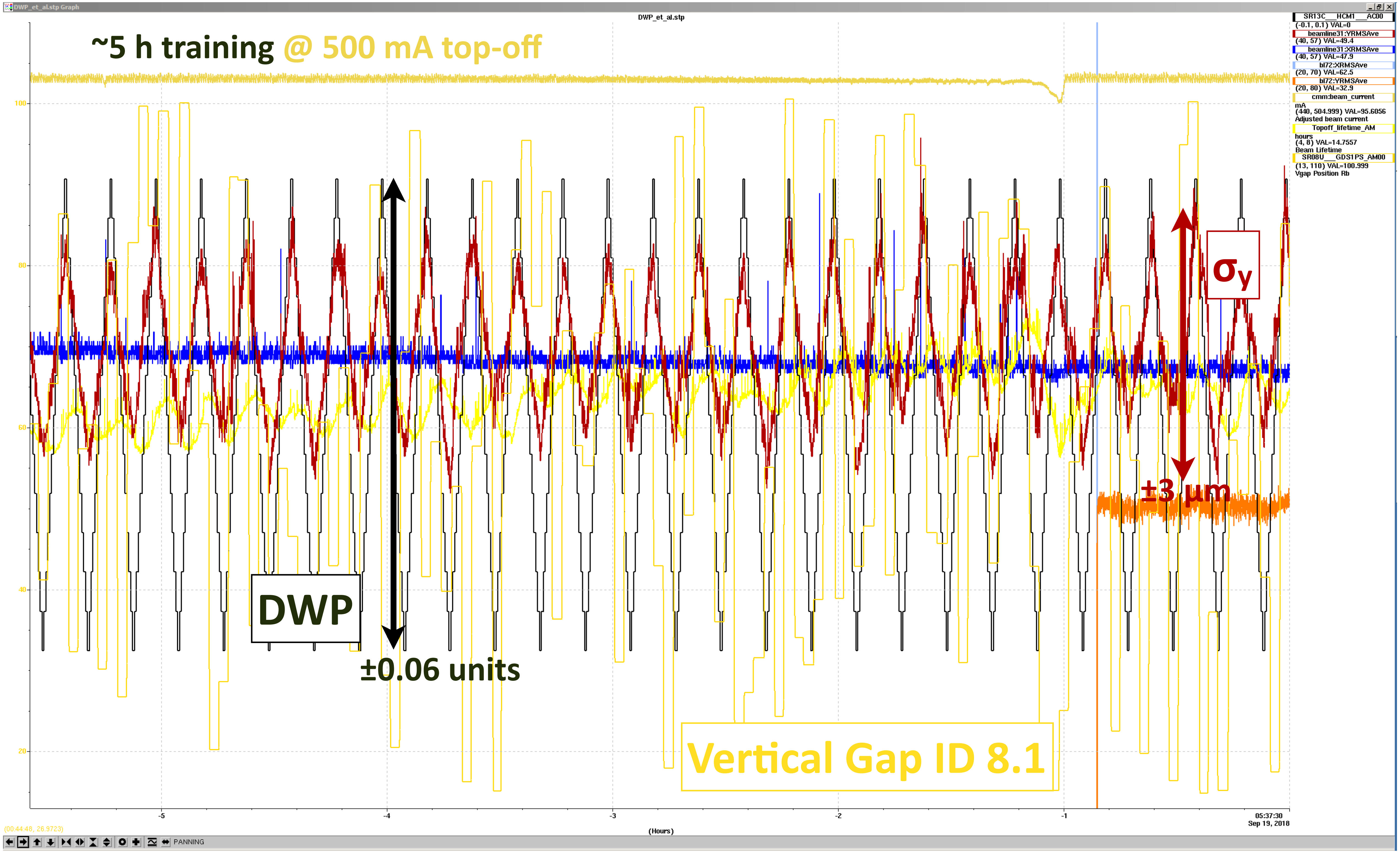


Courtesy: S. Liu

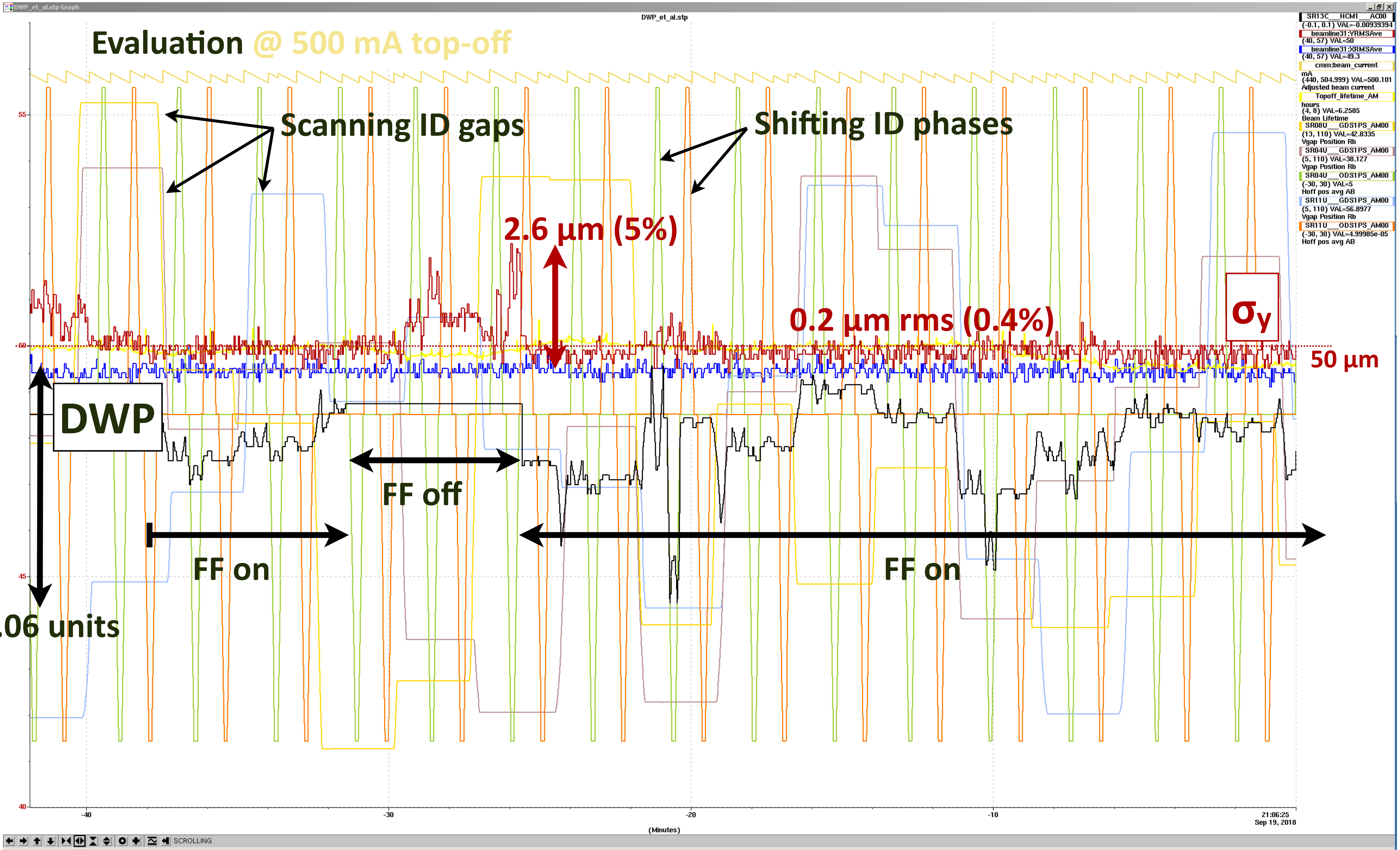
From prediction to correction—take 2

- Instead of dealing with FB issues, just scan DWP while scanning ID configurations → acquire data at 10 Hz → input for training of NN (DL)
- Result of DL is prediction for DWP required to keep beam size constant for arbitrary ID configurations
- Test case:
  - relied on 3 hours data acquisition (scanning 3 IDs)
  - 15 min to train NN (w/o special GPU)
  - then run skew correction based on DL prediction for DWP at 2 Hz

From prediction to correction—take 2



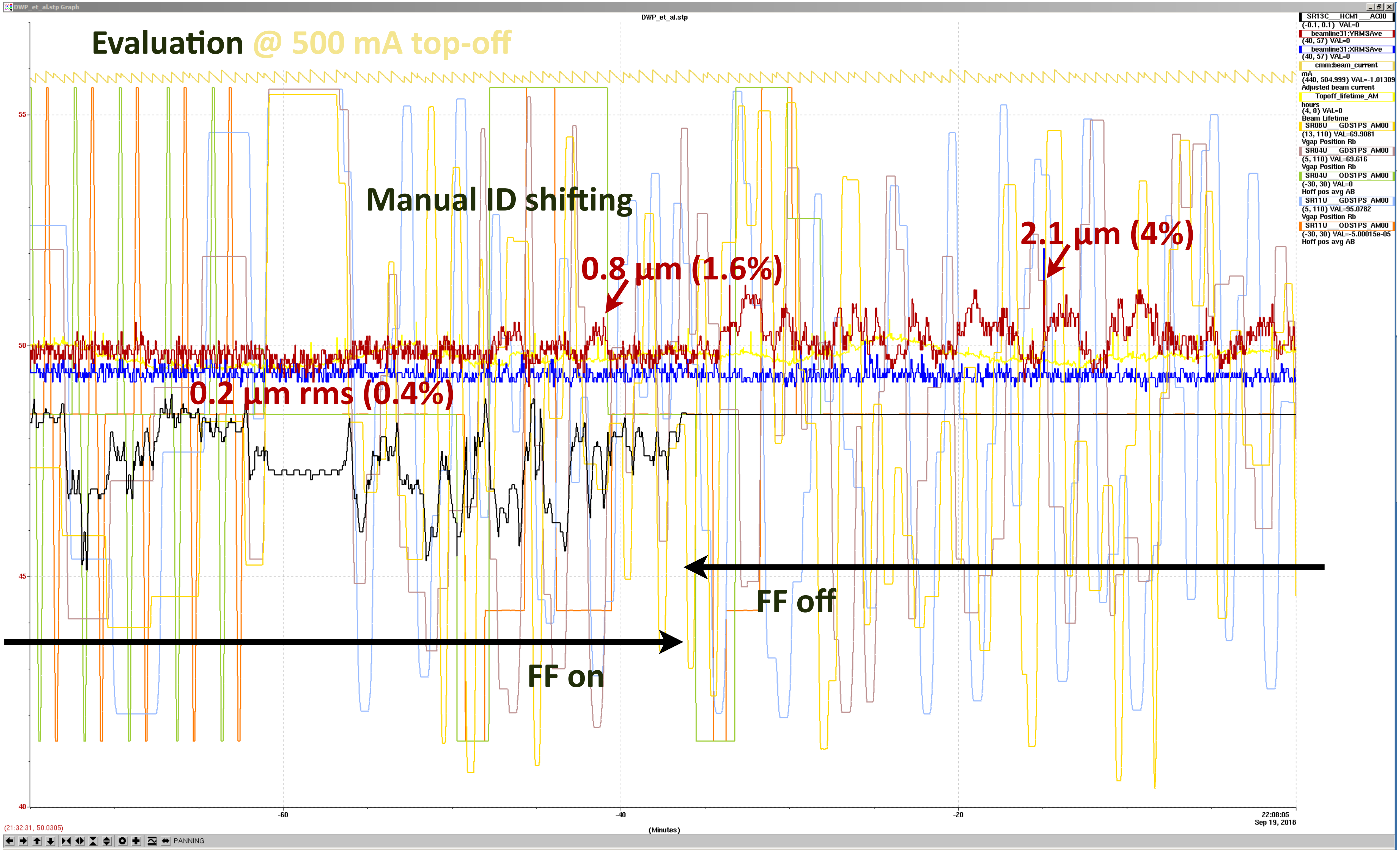
From prediction to correction—take 2



BERKELEY LAB

ALS

From prediction to correction—take 2



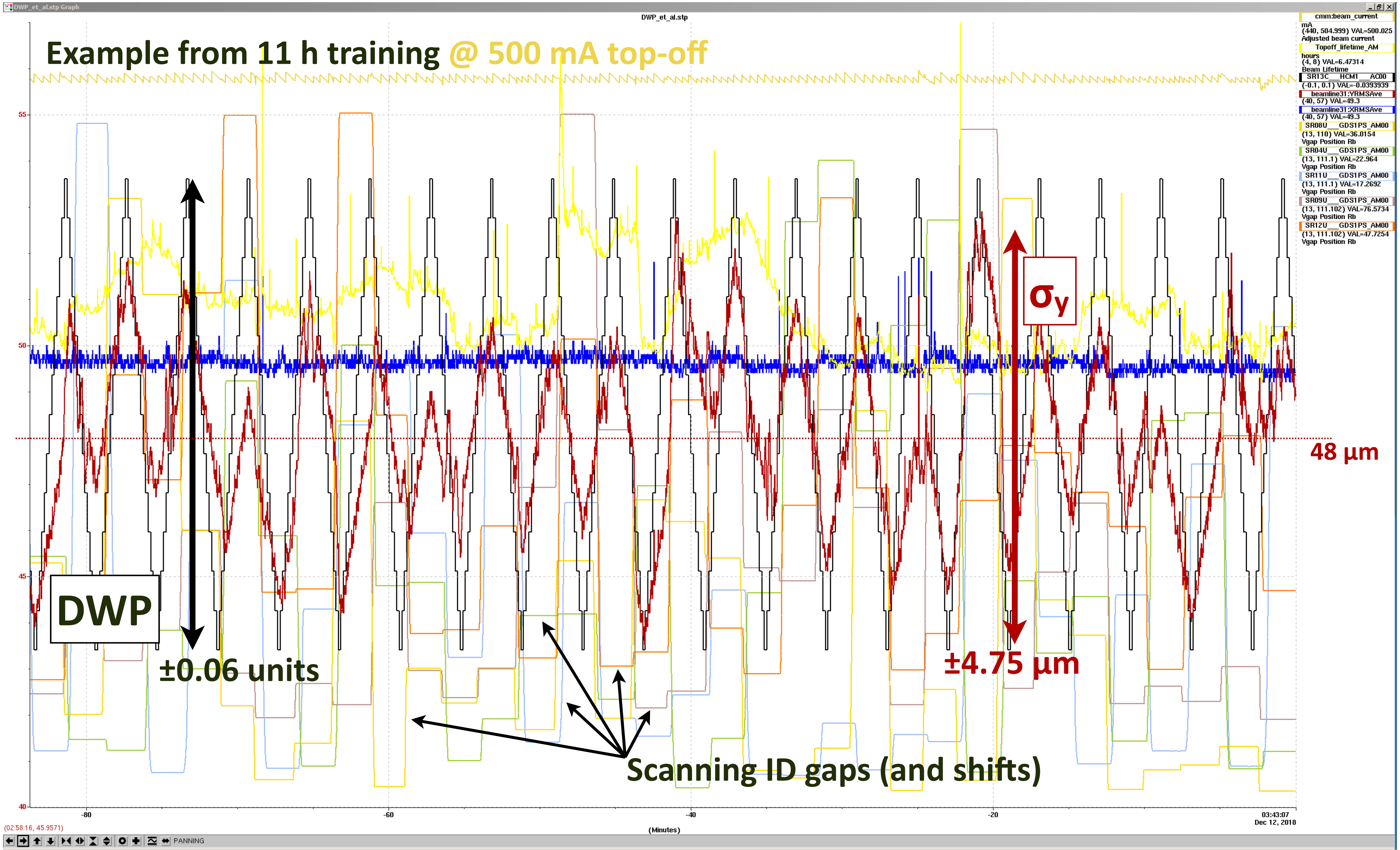
Fine, but does this thing really work?

- Scan more IDs, include all "worst offenders"
  - 3 EPUs, 2 planar undulators (4-1, 8, 9-1, 11, 12)
  - ramp across all shifts & scan full gap range (favor small gaps)

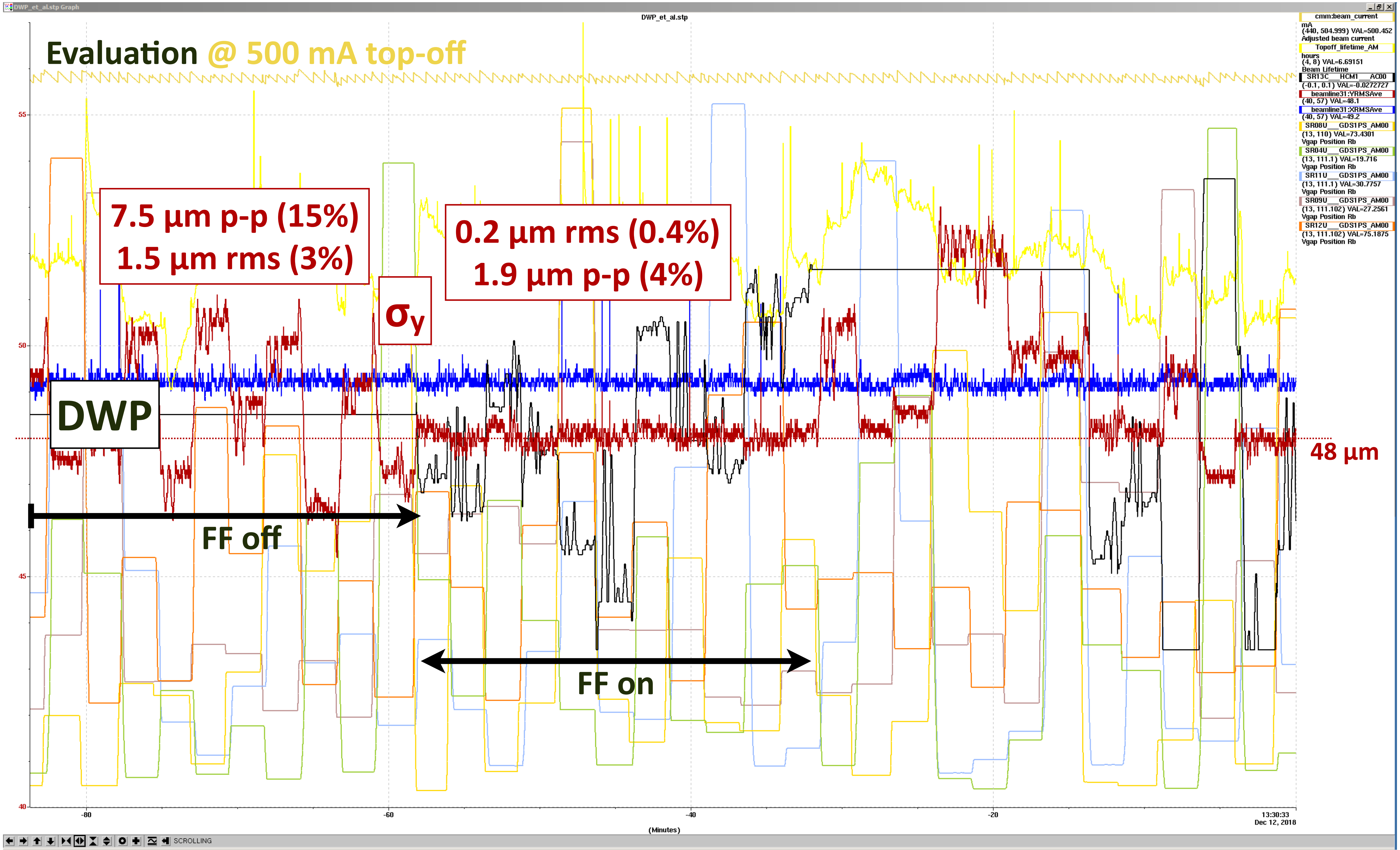
- Speed up BL 3.1 beam size measurement → speed up FF update rate

- Is stabilizing at BL 3.1 equivalent to stabilizing at user source points?
  - What are our most critical source points?
    → STXM BLs with monochromator dispersion in horizontal plane
  - BL 5.3.2.2 STXM scans at 1 ms/pixel, no independent concurrent I_0 measurement → cannot average out noise or normalize signal

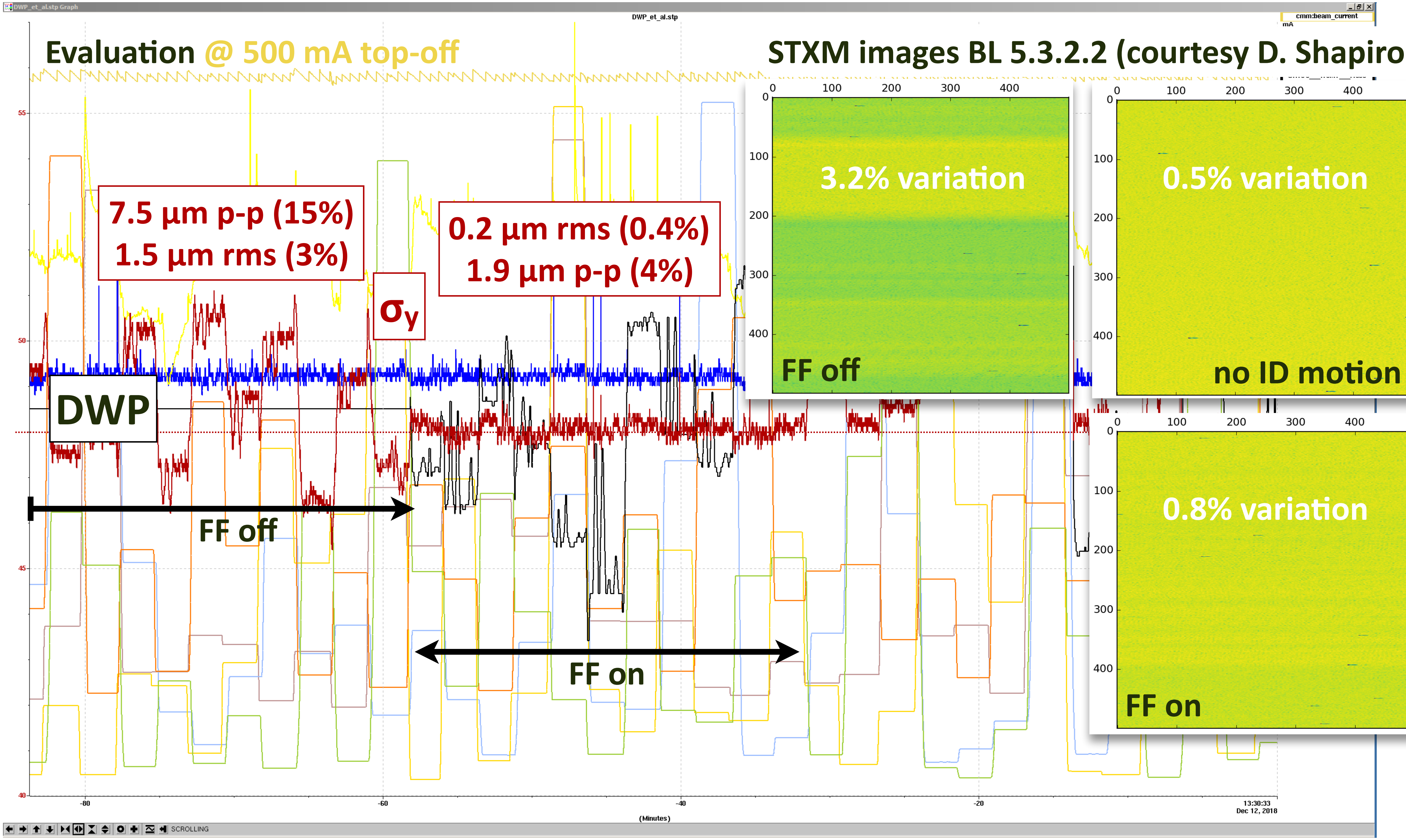
Fine, but does this thing really work?



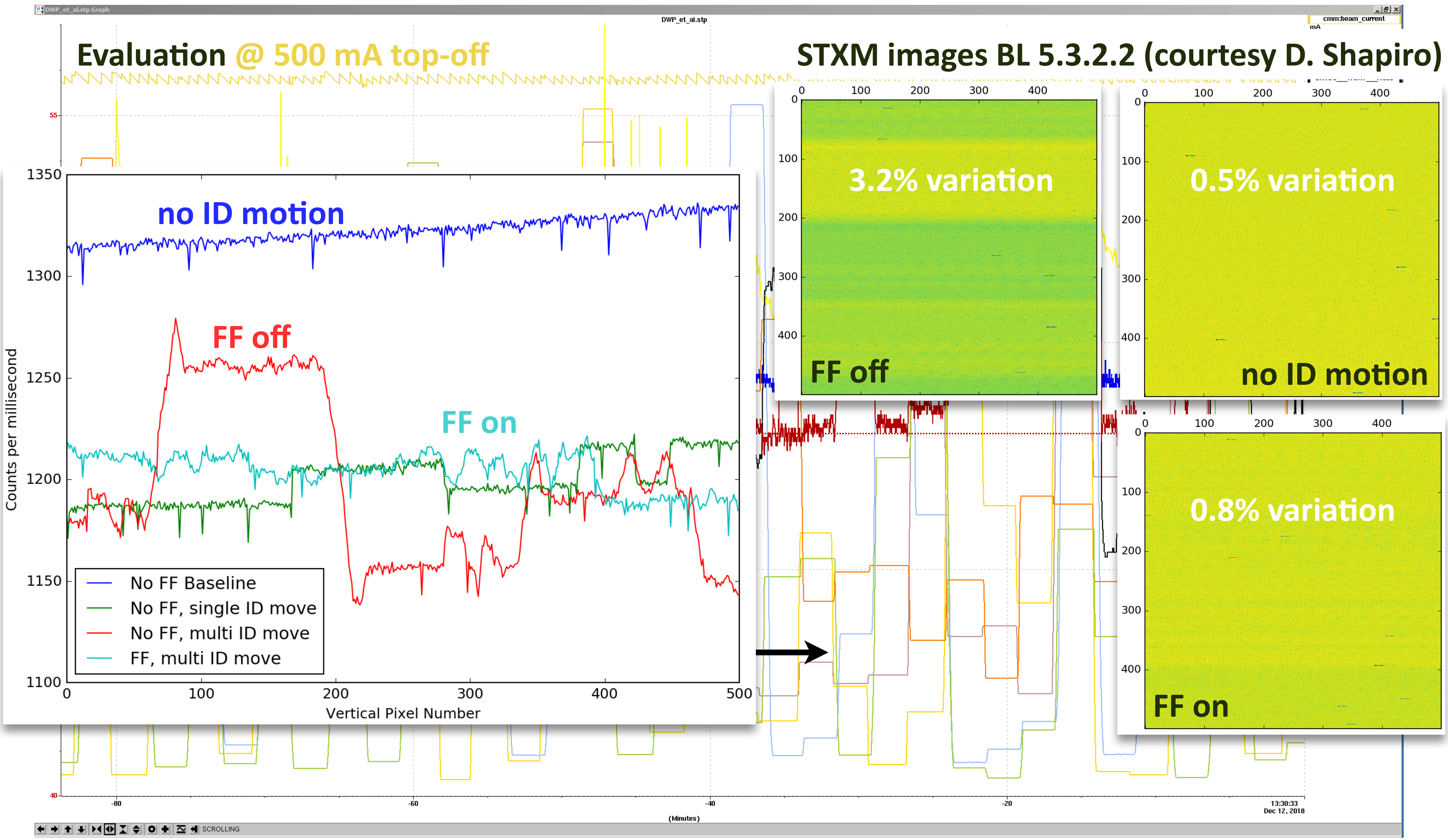
Fine, but does this thing really work?



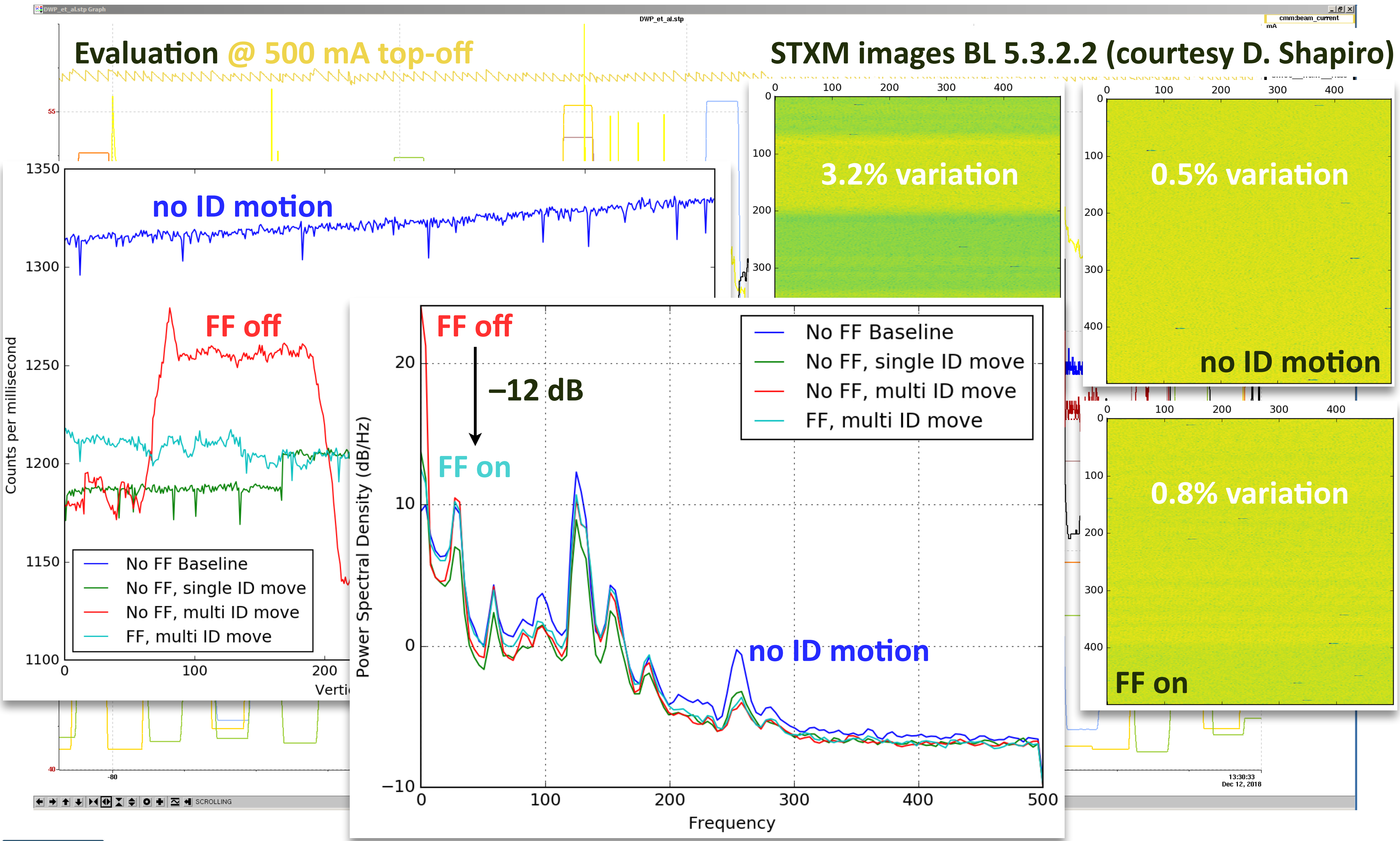
Fine, but does this thing really work?



Fine, but does this thing really work?

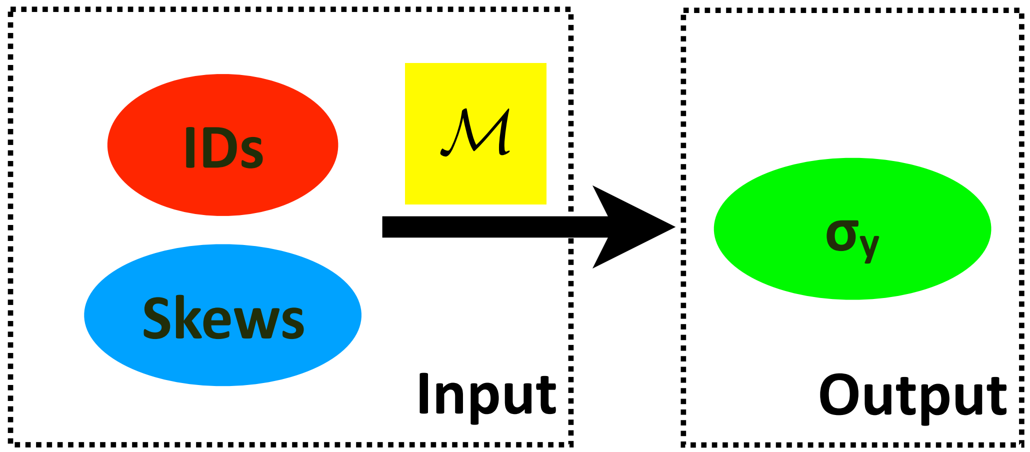


Fine, but does this thing really work?



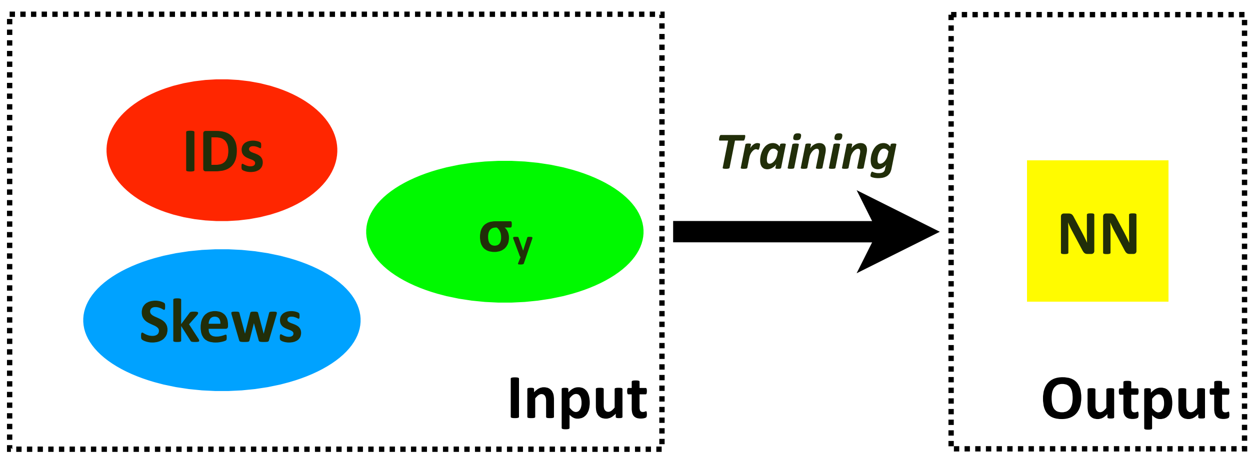
Deep Learning is a paradigm shift

Traditional Approach



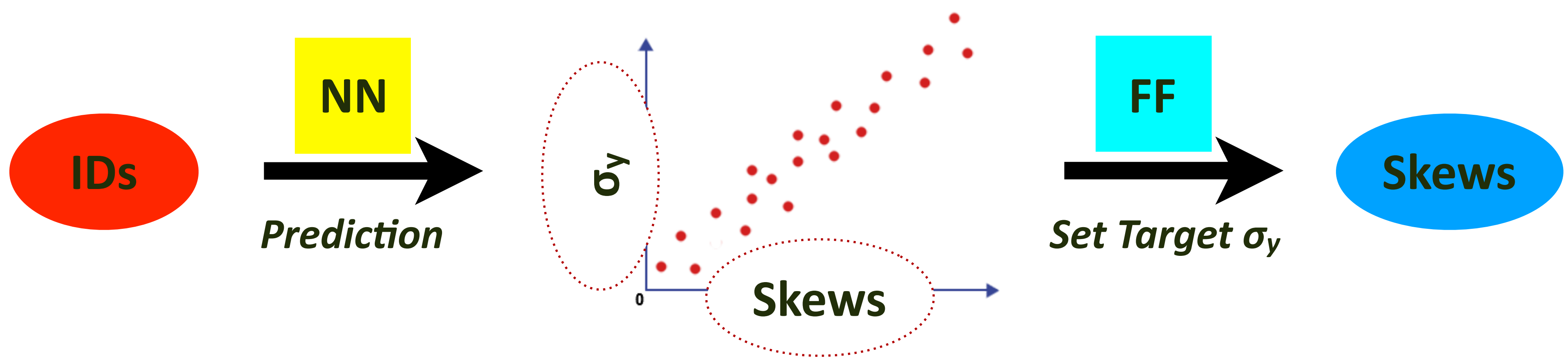
- *Requires physics/model knowledge*
- *Tends to rely on many (sometimes poor) assumptions*

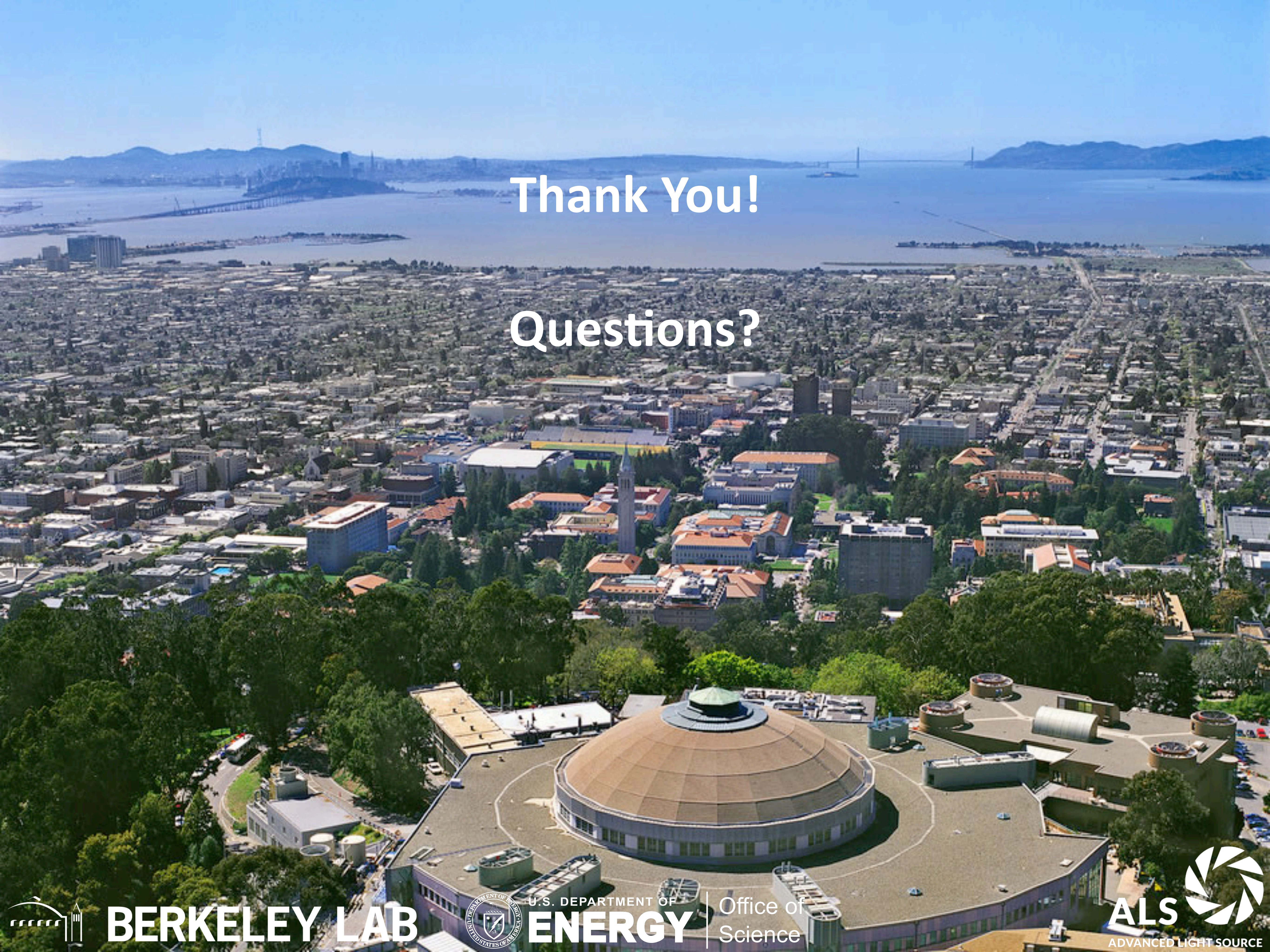
Deep Learning



- *Requires only large amounts of data & reproducibility*

Application during ops:





Thank You!

Questions?