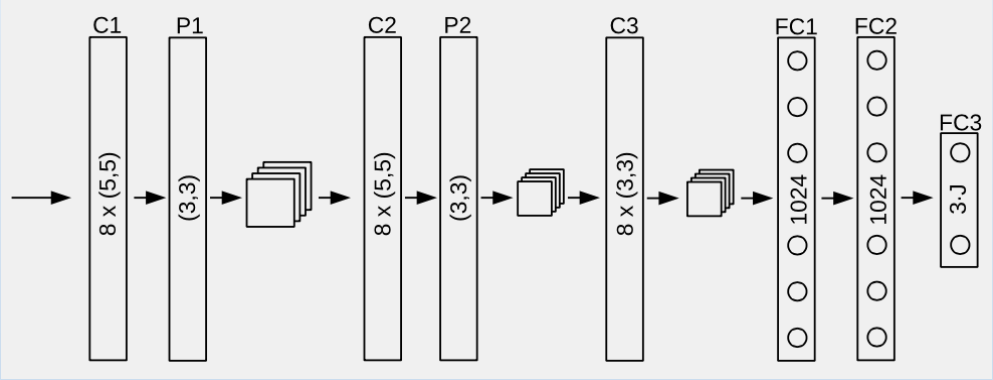
Imposing Hard Constraints on Deep Networks: Promises and Limitations



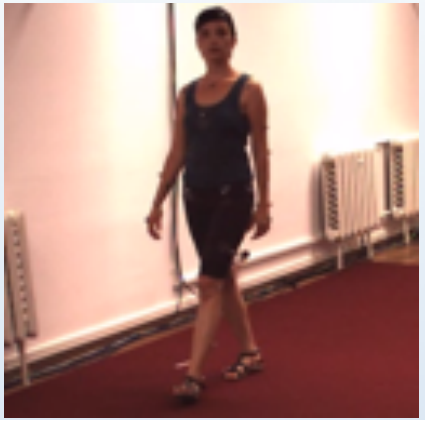
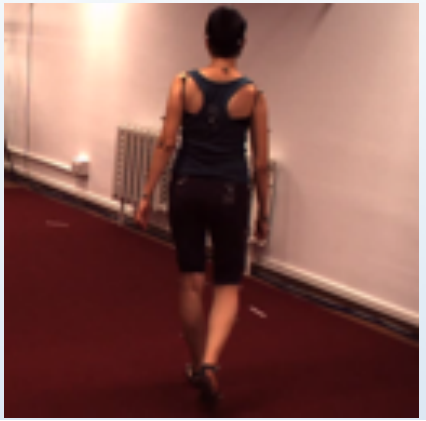
$$\min_{\mathbf{w}} R(\mathbf{w}) \text{ s.t } \mathbf{C}(\mathbf{w}) = 0$$

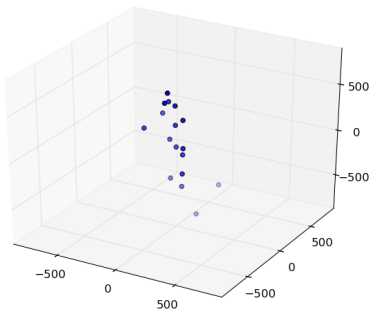
P. Marquez-Neila, M. Salzmann, and P. Fua

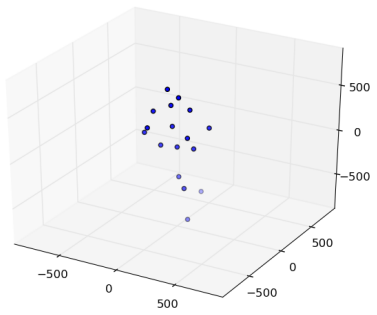
EPFL
Switzerland

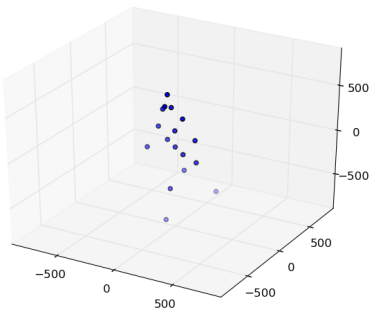
Motivation: 3D Pose Estimation

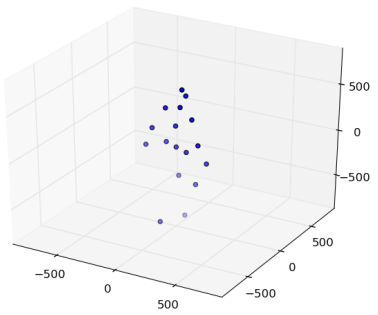






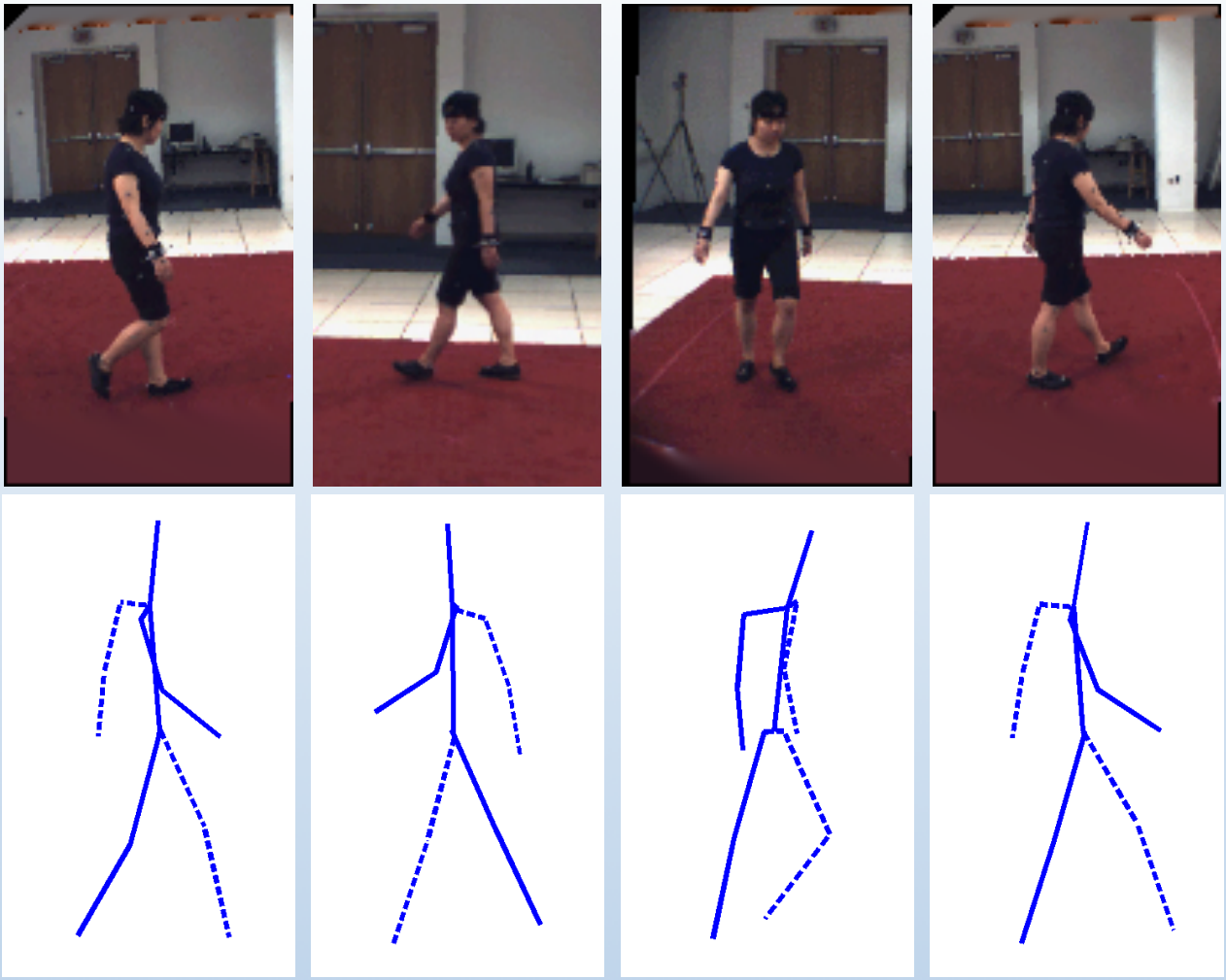


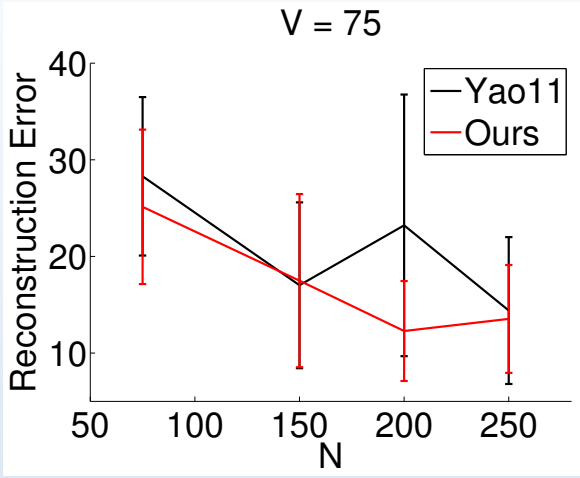


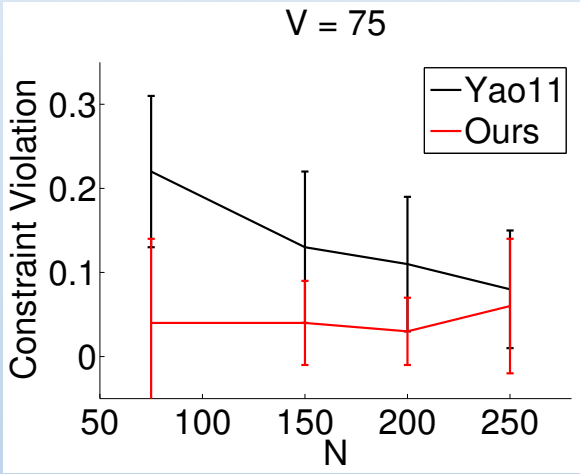
Given a CNN trained to predict the 3D locations of the person's joints:

- Can we increase precision by using our knowledge that her left and right limbs are of the same length?
- If so, how should such constraints be imposed?

In Shallower Times







Regression from PHOG features.

Constraining a Gaussian Latent Variable Model to preserve limb lengths:

- Better constraint satisfaction without sacrificing accuracy.
- Can this be repeated with CNNs?

Varol et al. CVPR'12

Standard Formulation

Given

- Deep Network architecture $\phi(\cdot, \mathbf{w})$,
- a training set $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i), 1 \leq i \leq N\}$ of N pairs of input and output vectors, $\mathbf{x}_i$ and $\mathbf{y}_i$,

find

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} R_{\mathcal{S}}(\mathbf{w}), \text{with } R_{\mathcal{S}}(\mathbf{w}) = \frac{1}{N} \sum_i L(\phi(\mathbf{x}_i, \mathbf{w}), \mathbf{y}_i) \ .$$

Adding Constraints

Hard Constraints:

Given a set of unlabeled points $\mathcal{U} = \{\mathbf{x}'_k\}_{k=1}^{|\mathcal{U}|}$, find

$$\min_{\mathbf{w}} R_{\mathcal{S}}(\mathbf{w}) \quad \text{s.t } C_{jk}(\mathbf{w}) = 0 \quad \forall j \leq N_C, \forall k \leq |\mathcal{U}| \ ,$$

where $C_{jk}(\mathbf{w}) = C_j(\phi(\mathbf{x}'_k; \mathbf{w}))$.

Soft Constraints:

Minimize

$$\min_{\mathbf{w}} R_{\mathcal{S}}(\mathbf{w}) + \sum_j \lambda_j \left(\sum_k C_{jk}(\mathbf{w})^2 \right) \ ,$$

where the λ_j parameters are positive scalars that control the relative contribution of each constraint.

In many "classical" optimization problems, hard constraints are preferred because they remove the need to adjust the λ values.

Lagrangian Optimization

Karush-Kuhn-Tucker (KKT) conditions:

$$\min_{\mathbf{w}} \max_{\Lambda} \mathcal{L}(\mathbf{w}, \Lambda), \text{ with } \mathcal{L}(\mathbf{w}, \Lambda) = R(\mathbf{w}) + \Lambda^T \mathbf{C}(\mathbf{w}),$$

Iterative minimization scheme:

$$\text{At each iteration } \mathbf{w} \leftarrow \mathbf{w} + \mathrm{d}\mathbf{w} \text{ with}$$

$$\begin{bmatrix} \mathbf{J}^T\mathbf{J} + \eta I & \frac{\partial \mathbf{C}}{\partial \mathbf{w}}^T \\ \frac{\partial \mathbf{C}}{\partial \mathbf{w}} & 0 \end{bmatrix} \begin{bmatrix} \mathrm{d}\mathbf{w} \\ \Lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{J}^T\mathbf{r}(\mathbf{w}_t) \\ -\mathbf{C}(\mathbf{w}_t) \end{bmatrix}$$

—> When there are millions of unknown these linear systems are **HUGE**!

Krylov Subspace Method

Solve

$$\mathbf{B}\mathbf{v} = \mathbf{b}$$

when the dimension of **v** so large that **B** cannot be stored in memory.

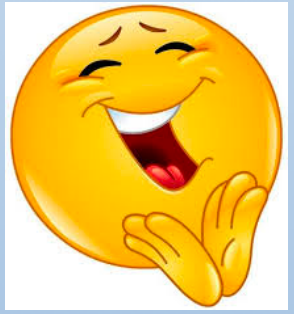
- Solve linear system by iteratively finding approximate solutions in the subspace spanned by

$$\{\mathbf{b}, \mathbf{B}\mathbf{b}, \mathbf{B}^2\mathbf{b}, ..., \mathbf{B}^k\mathbf{b}\} \text{ for } k = 0, 1, ..., N$$

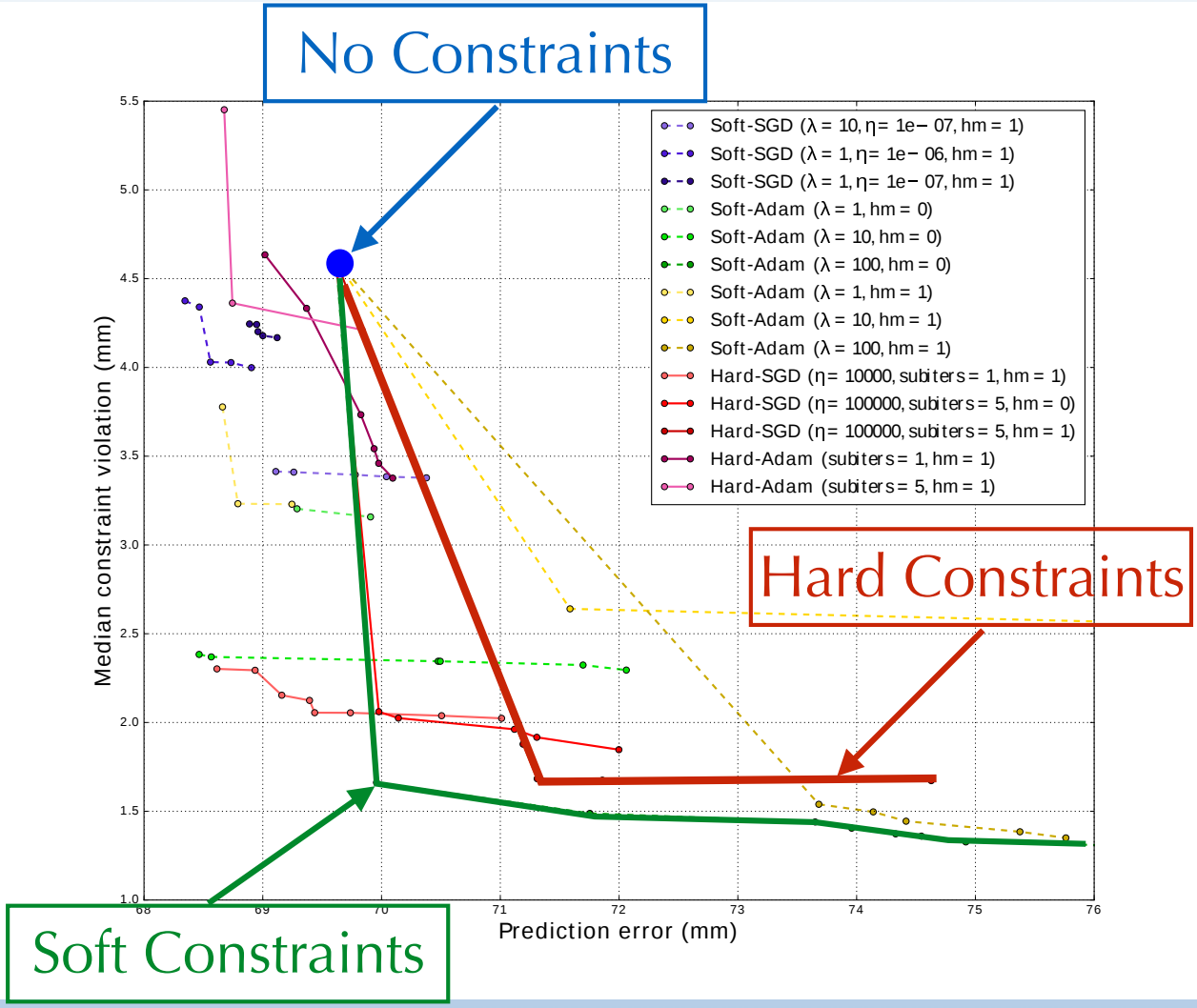
- Use *Pearlmutter* Trick to compute terms of the form

$$\mathbf{v}^T \frac{\partial \mathbf{f}}{\partial \mathbf{w}} \text{ and } \mathbf{v}^T \frac{\partial \mathbf{f}}{\partial \mathbf{w}}$$

—-> **It can be done!**

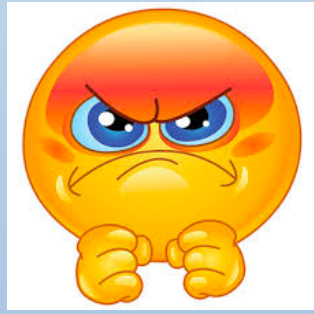


Results



Hard constraints help …

… but soft constraints help even more!



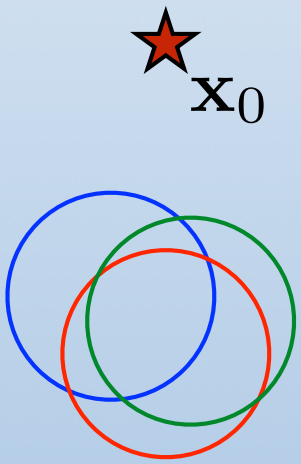
Synthetic Example

Let $\mathbf{x}$ and $\mathbf{c}_i$ for $1 \leq i \leq 200$ be vectors of dimension d and let $\mathbf{w}^*$ be either

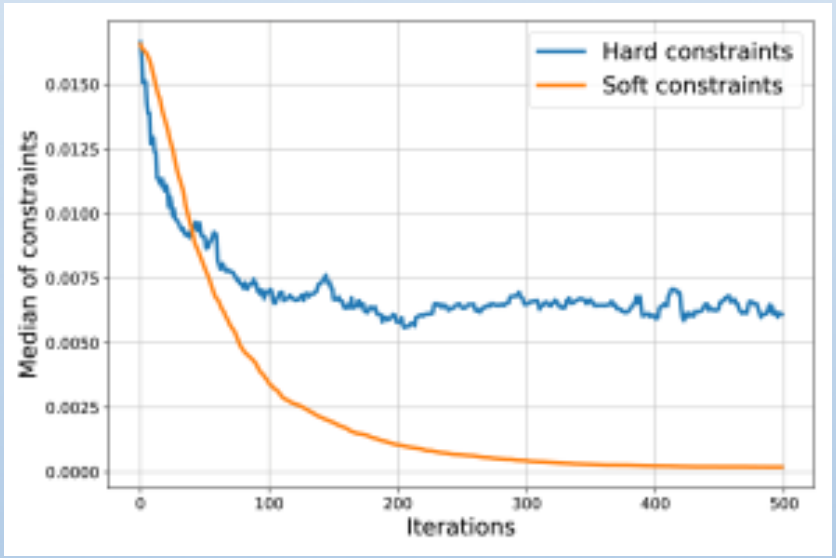
$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{x}_0\|^2 \text{ s.t } \|\mathbf{w} - \mathbf{c}_i\| - 10 = 0,\ 1 \leq i \leq 200 \qquad \text{(Hard Constraints)}$$

or

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{x}_0\|^2 + \lambda \sum_{1 \leq i \leq 200} \left(\|\mathbf{w} - \mathbf{c}_i\| - 10\right)^2 \qquad \text{(Soft Constraints)}$$

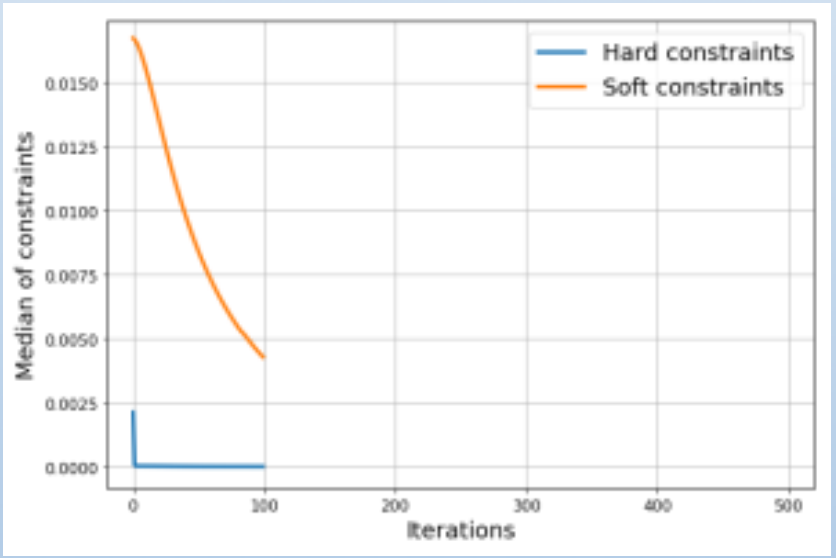


$d = 2$



$d = 1e6$

With constraint batches



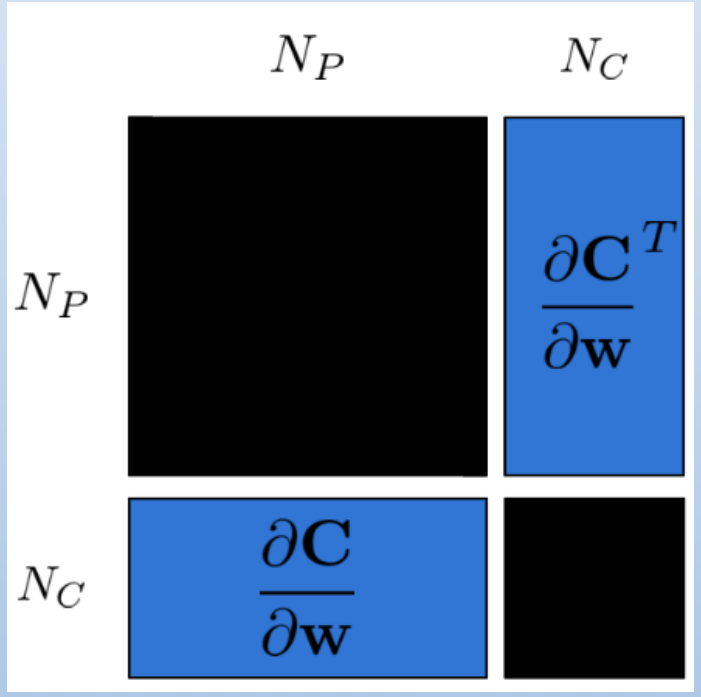
$d = 1e6$

All constraints active

Computational Complexity

At each iteration:

$$\begin{bmatrix} \mathbf{J}^T\mathbf{J} + \eta I & \frac{\partial \mathbf{C}}{\partial \mathbf{w}}^T \\ \frac{\partial \mathbf{C}}{\partial \mathbf{w}} & 0 \end{bmatrix} \begin{bmatrix} \mathrm{d}\mathbf{w} \\ \Lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{J}^T\mathbf{r}(\mathbf{w}_t) \\ -\mathbf{C}(\mathbf{w}_t) \end{bmatrix}$$



- Np is the size of a minibatch, which can be adjusted.
- Nc is the number of constraints, which is large if they are all active.

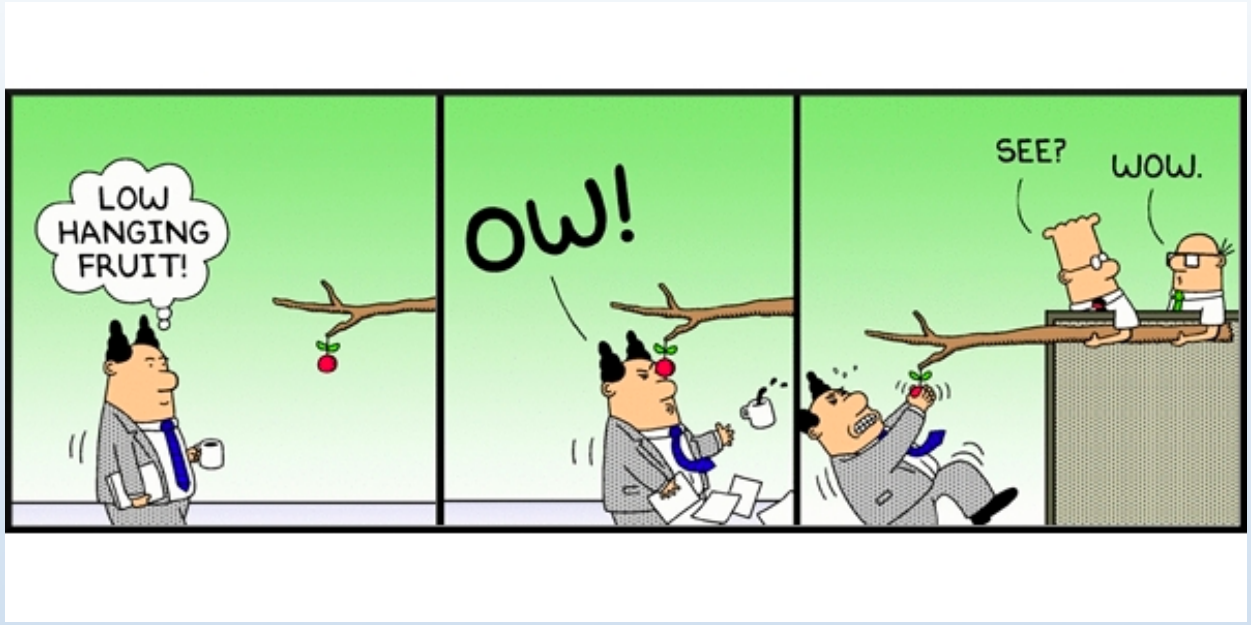
Interpretation

Hard constraints can be imposed on the output of Deep Nets but they are no more effective than soft ones:

- Not all constraints are independent from each other, which results in ill-conditioned matrices.
- We impose constraints on batches of constraints, which means we do not keep a consistent set of them.

—> We might still present this work at a positive result workshop.

Low Hanging Fruits



Typical approach:

- Identify an algorithm that can be naturally extended.
- Perform the extension.
- Show that your ROC curve is above the others.
- Publish in CVPR or ICCV.
- Iterate.

Shooting for the Moon

We choose to go to the Moon in this decade and do the other things, not because they are easy, but because they are hard.

J.F. Kennedy, 1962

In the context of Deep Learning:

- What makes Deep Nets tick?
- What are their limits?
- Can they be replaced by something more streamlined?