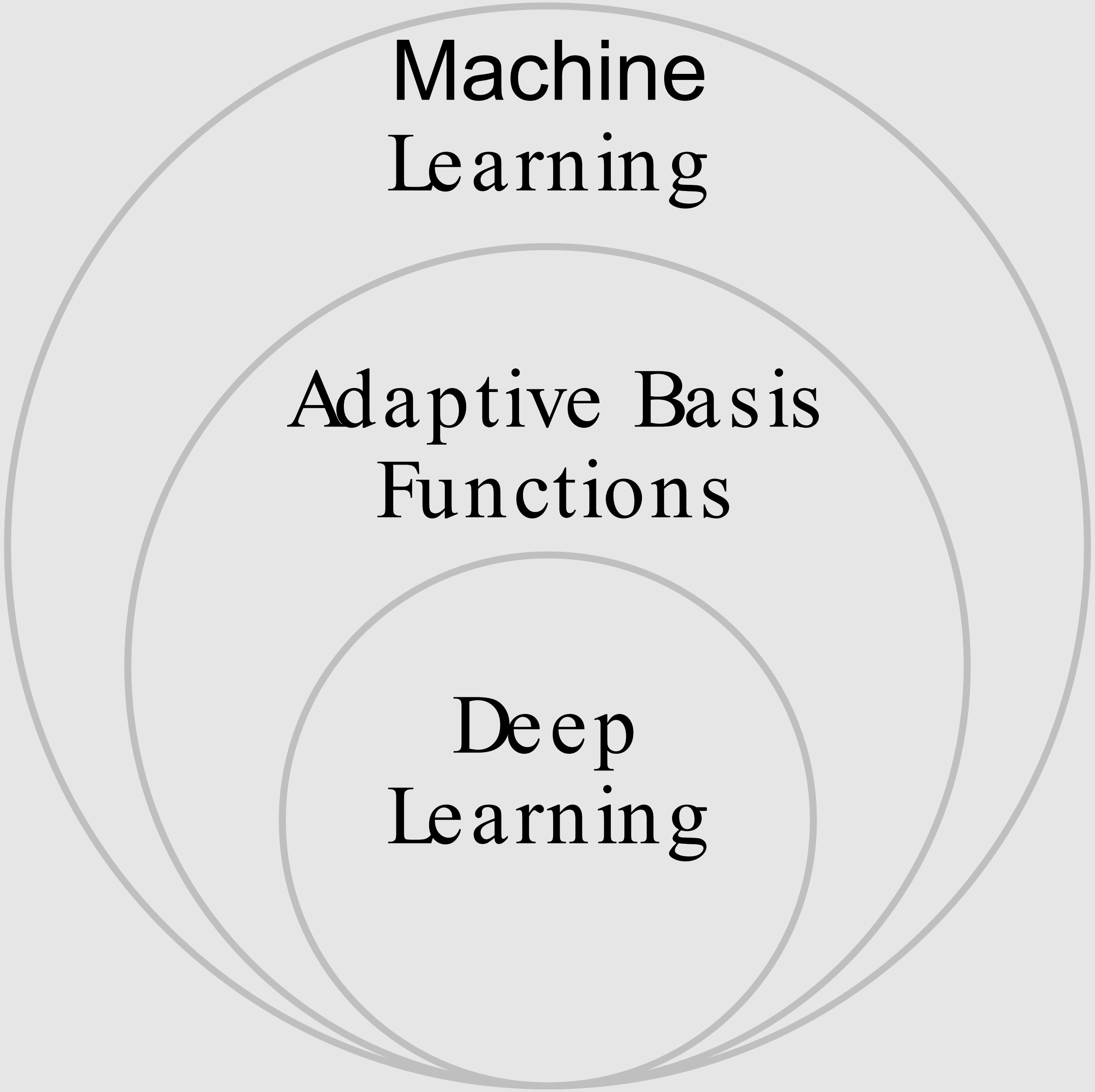
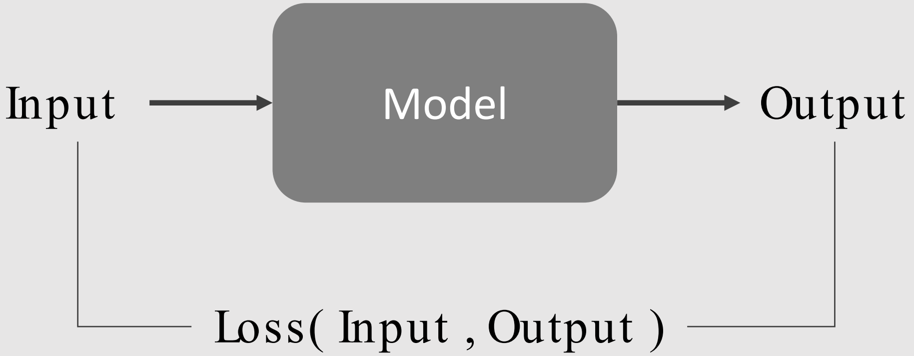
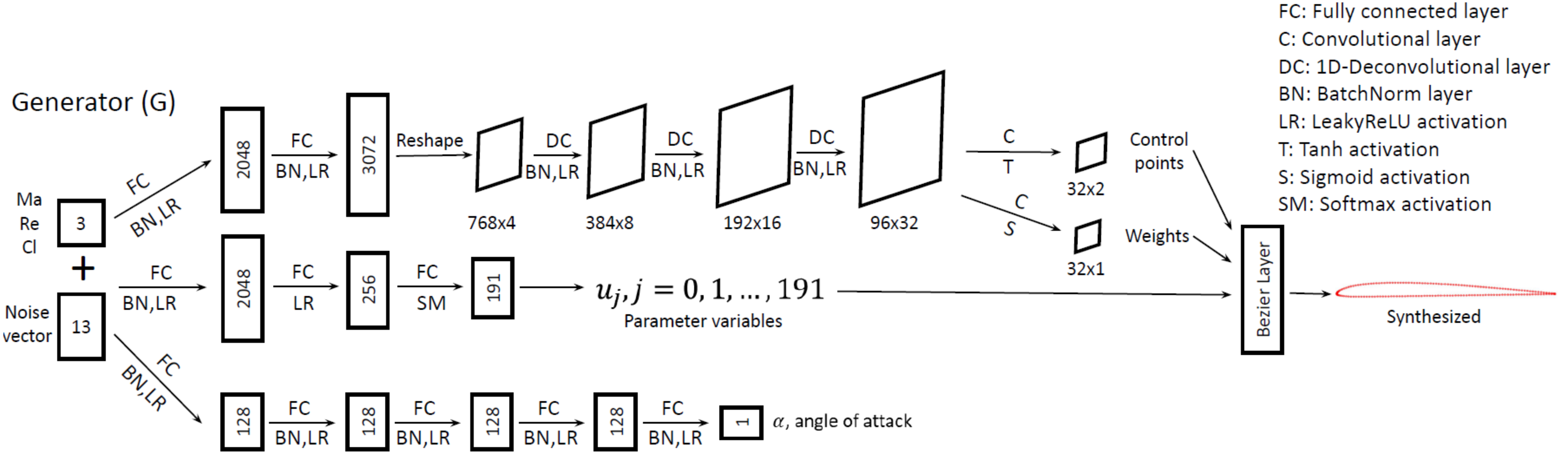
Introduction to Deep Learning

Dr. Mark Fuge
Univ. of Maryland, College Park
(301) 405-2558
fuge@umd.edu
ideal.umd.edu

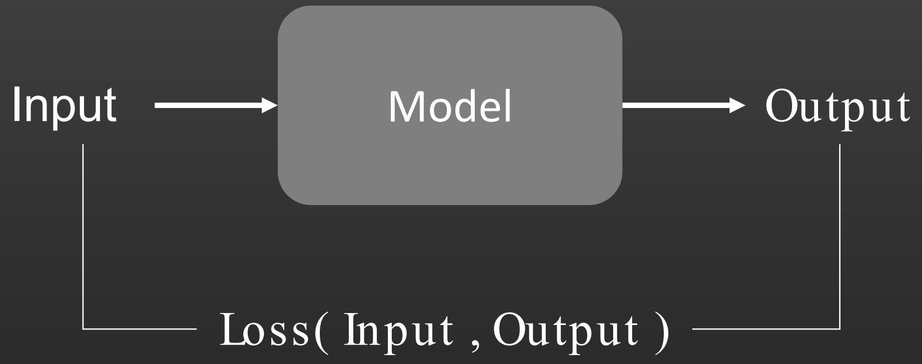

Machine
Learning
Adaptive Basis
Functions
Deep
Learning


Input
Model
Output
Loss(Input , Output)

Typical unreadable Deep Learning Slide from my research group for a recent DoE Technical Review



Let's build a Deep Learning model to predict airfoil lift



What should the input be?

(This will be our *basis function*)

Lift = Model(Input)

$$y = f(x)$$

How do you mathematically *represent* an airfoil?

How do you mathematically *represent* an airfoil?

$$\text{Lift} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{199} \\ w_{200} \end{pmatrix}^T \begin{pmatrix} x_1 \\ y_1 \\ \vdots \\ x_{100} \\ y_{100} \end{pmatrix}$$

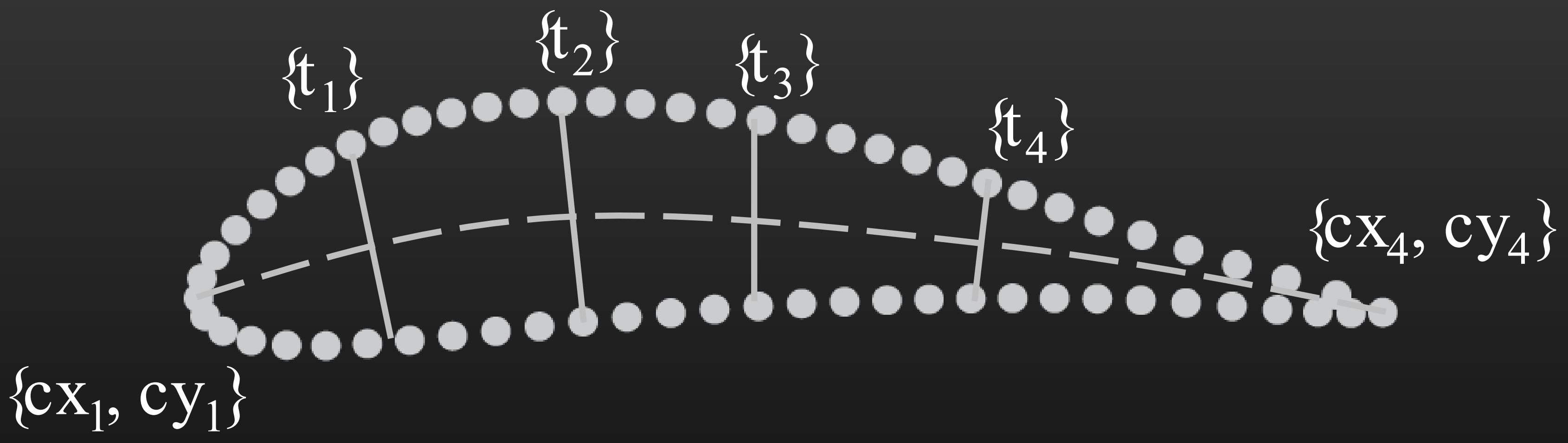
$$\text{Loss} = (\text{Lift}_{\text{predicted}} - \text{Lift}_{\text{actual}})^2$$
$$= (\ w^T x - \text{Lift}_{\text{actual}})^2$$

Find w where:

$$\partial \text{Loss} / \partial w = 0$$

How do you mathematically *represent* an airfoil?

$$\text{Lift} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{11} \\ w_{12} \end{pmatrix}^T \begin{pmatrix} cx_1 \\ cy_1 \\ \vdots \\ t_3 \\ t_4 \end{pmatrix}$$



Only thing we changed was the *basis*.

But the basis was fixed/ static.

What if we *adapted* or *learned* the basis?

How do you *adapt* a basis?

$$\text{Lift} = \begin{pmatrix} w_1 \\ w_2 \\ \\ \\ w_{199} \\ w_{200} \end{pmatrix}^T \begin{pmatrix} x_1 \\ y_1 \\ \\ \\ x_{100} \\ y_{100} \end{pmatrix}$$

$$\text{Loss} = (\text{Lift}_{\text{predicted}} - \text{Lift}_{\text{actual}})^2$$
$$= (w^T x - \text{Lift}_{\text{actual}})^2$$

How do you *adapt* a basis?

$$\text{Lift} = \begin{pmatrix} w_1 \\ w_2 \\ \\ \\ \\ w_{199} \\ w_{200} \end{pmatrix}^T \begin{pmatrix} g(x_1) \\ g(y_1) \\ \\ \\ \\ g(x_{100}) \\ g(y_{100}) \end{pmatrix}$$

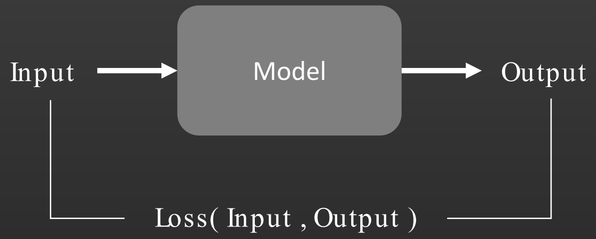
$$\text{Loss} = (\text{Lift}_{\text{predicted}} - \text{Lift}_{\text{actual}})^2$$
$$= (w^T g(x) - \text{Lift}_{\text{actual}})^2$$

How do you *adapt* a basis?

$$\text{Lift} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{199} \\ w_{200} \end{pmatrix}^T \begin{pmatrix} g(x_1) \\ g(y_1) \\ \vdots \\ g(x_{100}) \\ g(y_{100}) \end{pmatrix}$$

What is g?

Another model!



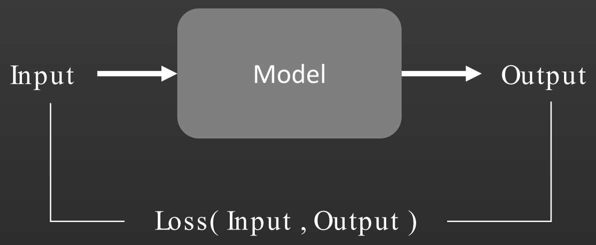
$$\text{Loss} = (\text{Lift}_{predicted} - \text{Lift}_{actual})^2$$
$$= (w^T g(x) - \text{Lift}_{actual})^2$$
$$= (w_1^T\{w_2^T(x)\} - \text{Lift}_{actual})^2$$

How do you *adapt* a basis?

$$\text{Lift} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{199} \\ w_{200} \end{pmatrix}^T \begin{pmatrix} g(x_1) \\ g(y_1) \\ \vdots \\ g(x_{100}) \\ g(y_{100}) \end{pmatrix}$$

What is g?

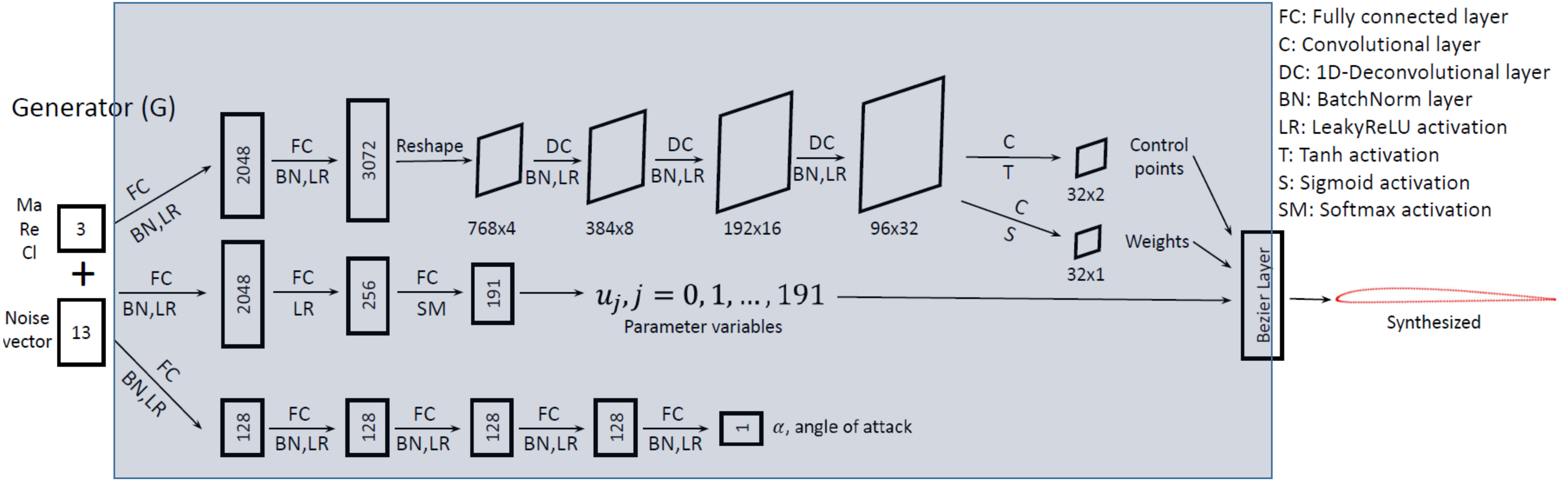
Another model!



$$\begin{aligned}\text{Loss} &= (\text{Lift}_{predicted} - \text{Lift}_{actual})^2 \\ &= (\; w^T g(x) \; - \text{Lift}_{actual})^2 \\ &= (\sigma(w_1^T\{\sigma(w_2^T(x))\}) - \text{Lift}_{actual})^2\end{aligned}$$

Now we can see that the Deep Learning model is (in essence) a series of chained basis transformations!

Model g(x)



FC: Fully connected layer
C: Convolutional layer
DC: 1D-Deconvolutional layer
BN: BatchNorm layer
LR: LeakyReLU activation
T: Tanh activation
S: Sigmoid activation
SM: Softmax activation

Why use Deep Learning over other (non-Deep) approaches or not?

Advantages

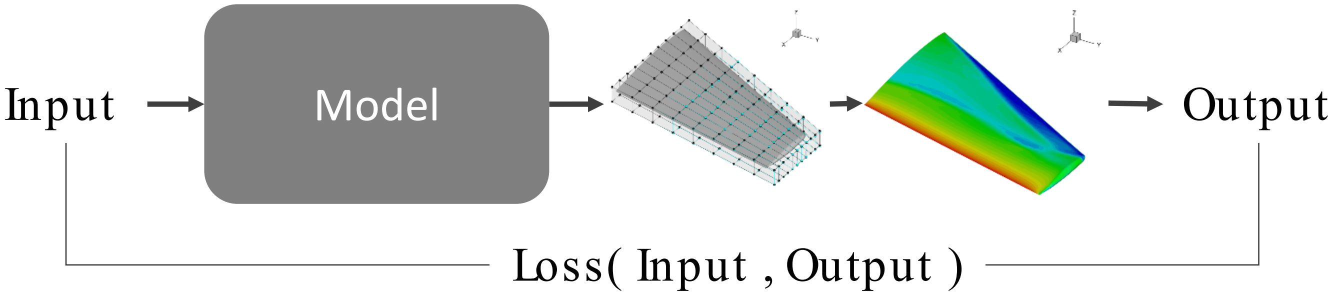
1. Fairly extensible with modern libraries
2. Plays nicely with other differentiable approaches
3. Good hardware acceleration
4. Active research community + industrial investment

Disadvantages

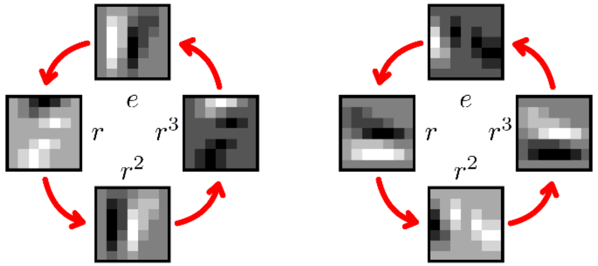
1. Certain modeling assumptions difficult to do
2. Certain architectures have difficulty converging or possess pathologies
3. Theory less developed than some other models

Opportunities and Directions

Merging of Engineering and Deep-Learning models



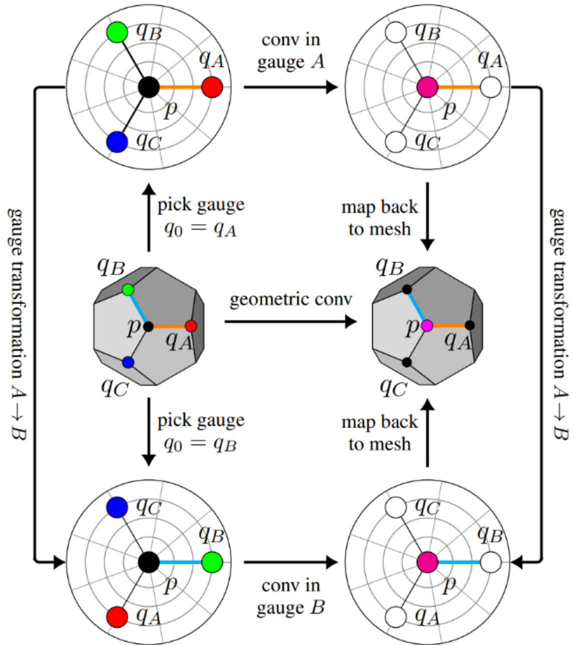
New invariances & constraints on Deep Learning models



Cohen and Welling, 2016
arxiv.org/ abs/ 1602.07576

Generalizing Convolution

de Haan et al., 2020
arxiv.org/ abs/ 2003.05425



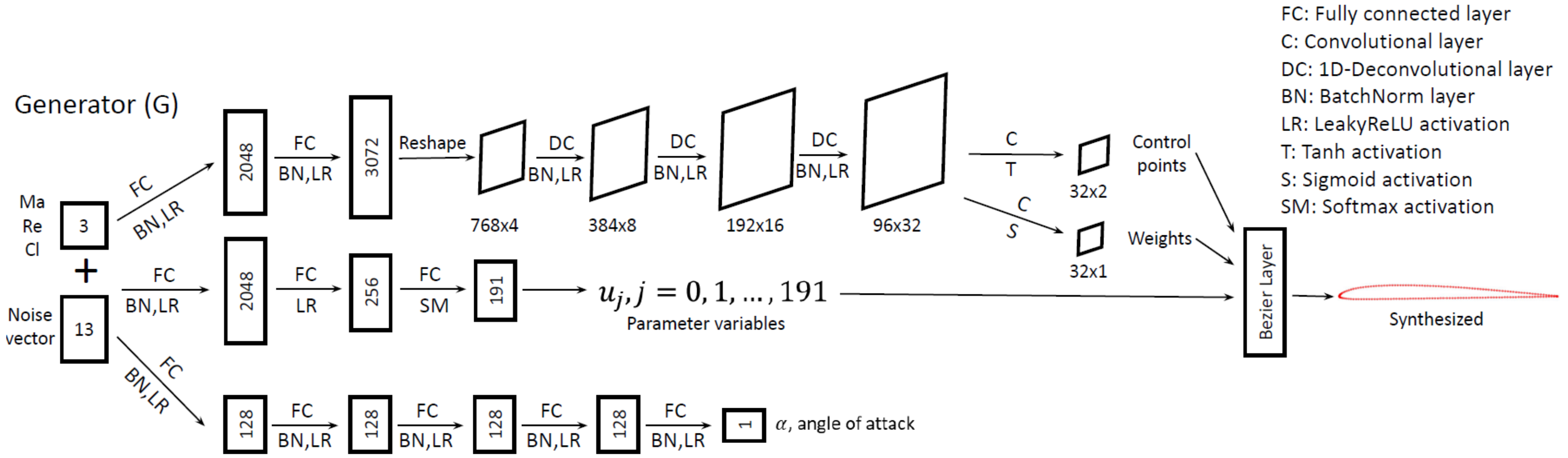
Combining Probabilistic and Deep Learning Models

Thank you

Dr. Mark Fuge
Univ. of Maryland, College Park
(301) 405-2558
fuge@umd.edu
ideal.umd.edu

Backup Slides

Now we can see that the Deep Learning model is (in essence) a series of chained basis transformations!



Minimize Sinkhorn divergence:

$$\min_G S_\lambda(p_r(\boldsymbol{x},\boldsymbol{y}) \| p_g(\boldsymbol{x}|\boldsymbol{y}) p_r(\boldsymbol{y}))$$

Loss

Conditional Formulation:

$$p(\mathbf{x},\mathbf{y}) = \int_{\hat{\mathbf{y}},\mathbf{z}} p(\mathbf{x},\mathbf{y} \mid \hat{\mathbf{y}},\mathbf{z})\, p_r(\hat{\mathbf{y}})\, p(\mathbf{z})\, d\hat{\mathbf{y}}\, d\mathbf{z}$$

$$p(\mathbf{x},\mathbf{y} \mid \hat{\mathbf{y}},\mathbf{z}) \propto \exp - \frac{c([\mathbf{x},\mathbf{y}],[G(\mathbf{z},\hat{\mathbf{y}}),\hat{\mathbf{y}}])}{\lambda}.$$

Surrogate Log-Likelihood (SLL):

$$\log p(\mathbf{x},\mathbf{y}) \geq -\frac{1}{\lambda}\mathbb{E}_{\mathbb{P}^\star_{Z|X,Y}}\left[c([\mathbf{x},\mathbf{y}],[G^\star(\mathbf{z},\mathbf{y}),\mathbf{y}])\right]$$
$$+\mathbb{E}_{\mathbb{P}_Z}[\log p(\mathbf{z})] + H\left(\mathbb{P}^\star_{Z|X,Y}\right) + \log p_r(\mathbf{y}) + \text{const}$$

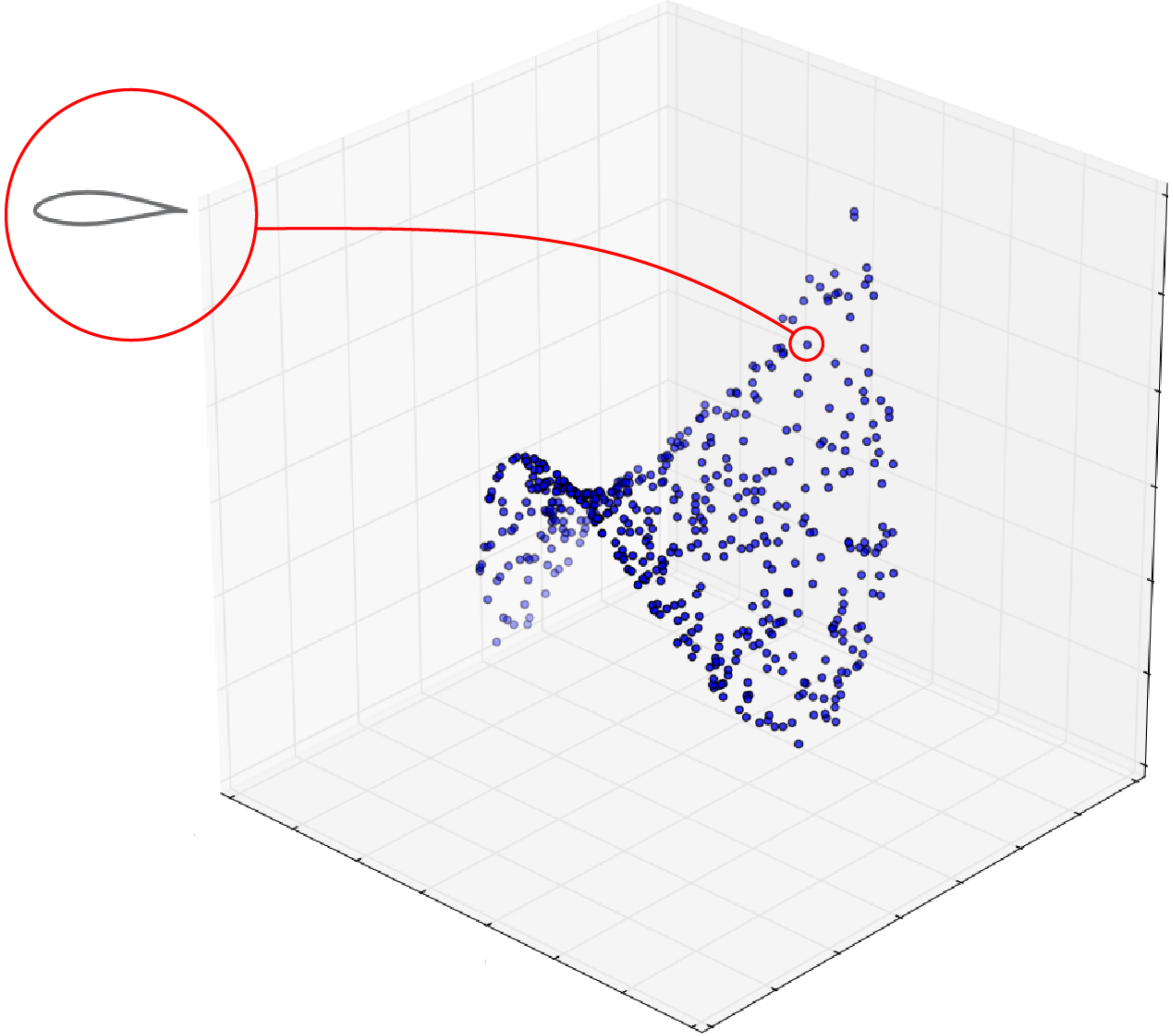
in which

$$\mathbb{P}^\star_{Z|X,Y} = \mathbb{P}_Z(\mathbf{z}) \exp \frac{v^\star([\mathbf{x},\mathbf{y}],[G^\star(\mathbf{z},\mathbf{y}),\mathbf{y}])}{\lambda}.$$

Cost Function:

$$c([\mathbf{x},\mathbf{y},\mathbf{b}],[\hat{\mathbf{x}},\hat{\mathbf{y}},\hat{\mathbf{b}}]) = |\mathbf{x}-\hat{\mathbf{x}}| + |\mathbf{y}-\hat{\mathbf{y}}| + |\mathbf{b}-\hat{\mathbf{b}}|$$

What are Generative Models doing?



$$f: \mathcal{Z} \to \mathcal{X} \qquad \mathbb{P}(\boldsymbol{x}|\boldsymbol{z})$$

$$f^{-1}: \mathcal{X} \to \mathcal{Z} \qquad \mathbb{P}(\boldsymbol{z}|\boldsymbol{x})$$

$$\log \mathbb{P}(\boldsymbol{x}) = \log \mathbb{P}(\boldsymbol{z}) + \log|\det \nabla_{\boldsymbol{x}} f^{-1}(\boldsymbol{x})|$$