Visual Anagrams and Applications

Ashwin Baluja, 4/18/24

cvpr 2024

Visual Anagrams: Generating Multi-View Optical Illusions with Diffusion Models

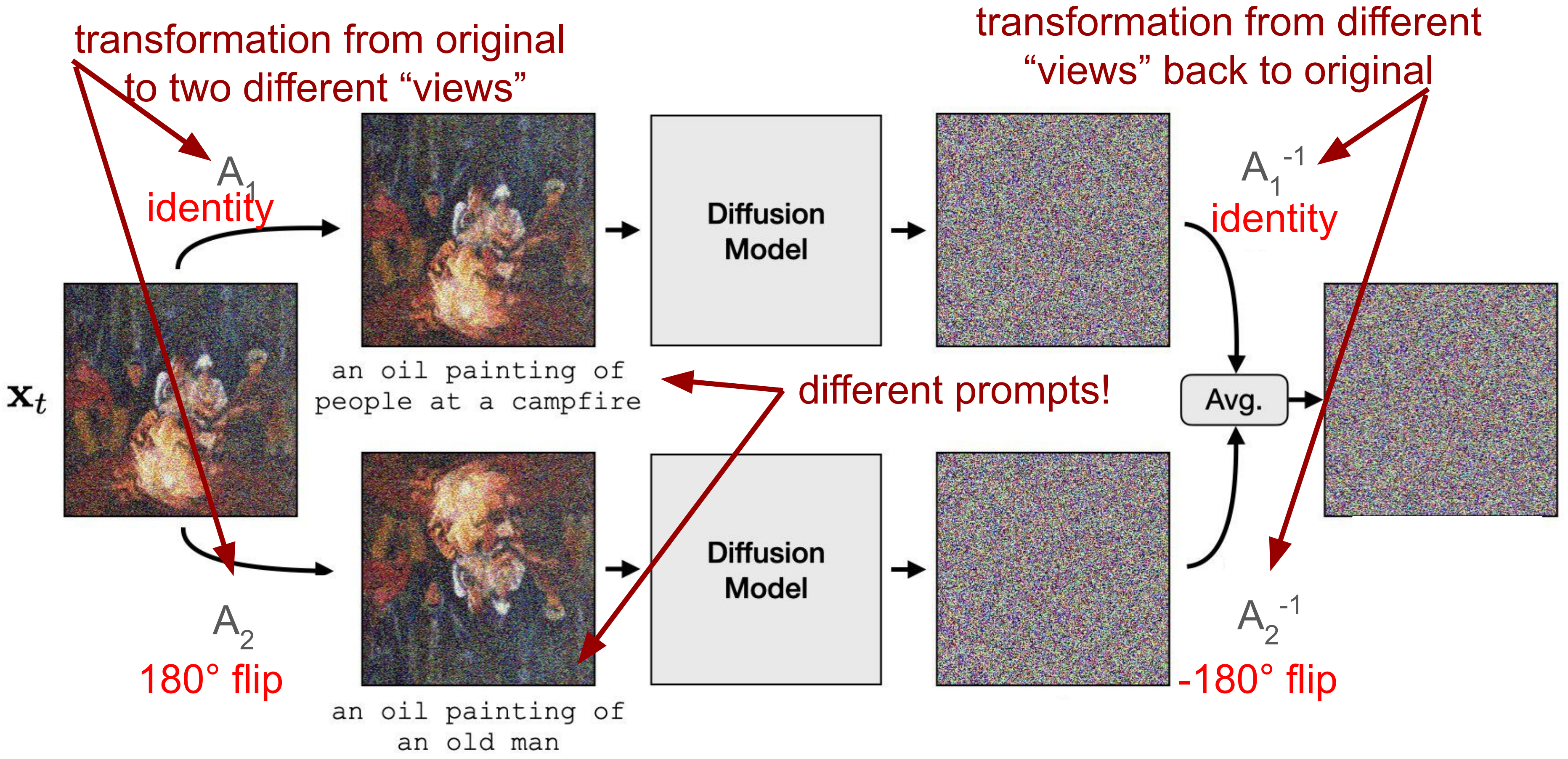
Daniel Geng, Inbum Park, Andrew Owens
University of Michigan
Correspondence to: dgeng@umich.edu

a drawing
of a penguin

a drawing
of a giraffe

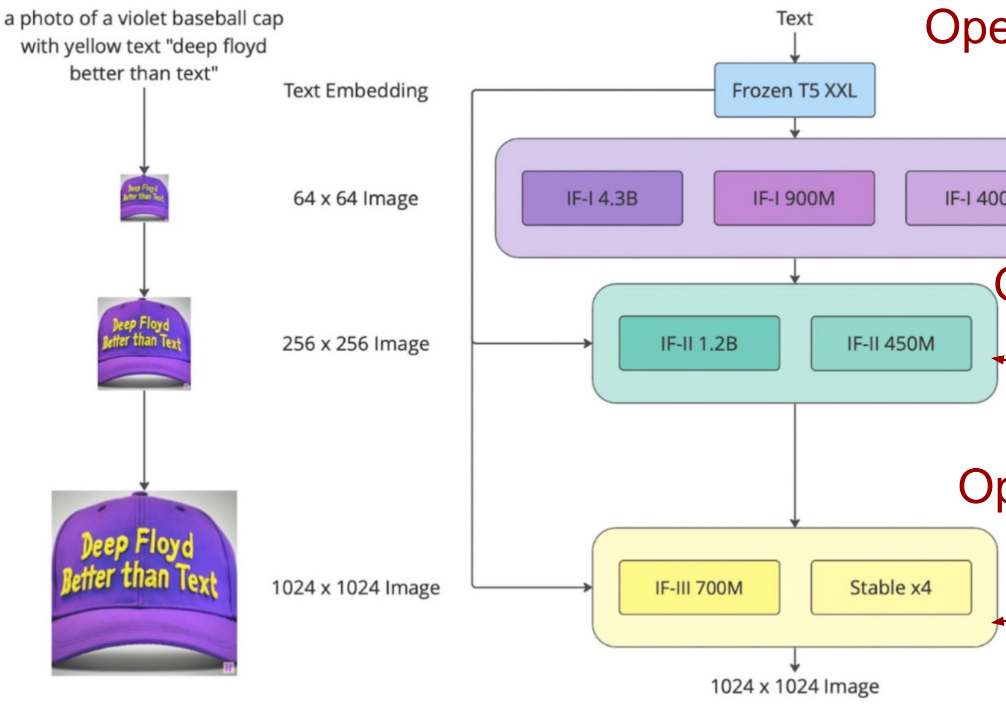
an oil painting
of a horse

an oil painting of
a snowy mountain village



transformation from original to two different "views"
transformation from different "views" back to original
A_1
identity
A_2
180° flip
$\mathbf{x}_t$
an oil painting of people at a campfire
an oil painting of an old man
different prompts!
Diffusion Model
Diffusion Model
A_1^{-1}
identity
A_2^{-1}
-180° flip
Avg.

We need a direct relation between output pixels and noise



Operates on 64x64 image

Operates on 256x256 image

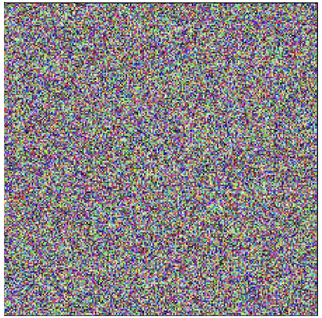
Operates on 1024x1024 image

What types of transformations work?

Expected input for diffusion model

Signal

$+\ a_t$



Noise

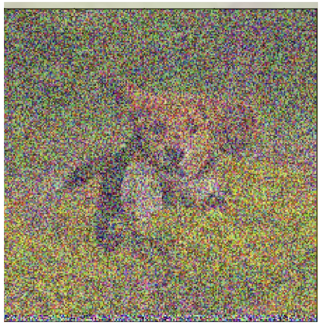
transformation between different "views"

$$A\left(\textbf{Signal} + a_t(\textbf{Noise})\right) = A\left(\textbf{Signal}\right) + A\left(a_t\ \textbf{Noise}\right)$$

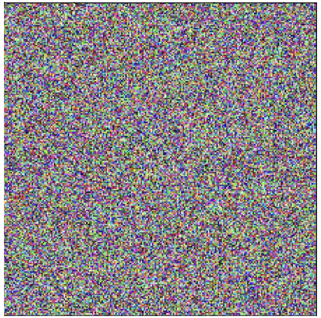
A must be linear!

transformation back to original view

What types of transformations work?



Diffusion model tries to predict noise



$\} \mathcal{N}(0, 1)$

A (our transformation) must preserve $\mathcal{N}(0, 1)$

$A(\text{pred\_noise}) \sim \mathcal{N}(0, 1) \Longrightarrow \mathrm{Cov}(A(\text{pred\_noise})) = \mathrm{I}$

(given that mean = 0) $\mathrm{Cov}(A(\text{pred\_noise})) = AA^T$

$AA^T \Longrightarrow$ A is orthogonal matrix

What types of transformations work?

A is a linear matrix that is orthogonal

$$\overbrace{}$$

“flips, rotations, skews, color inversions, and jigsaw rearrangements”

“any orthogonal transformation works as a view with our method”

View: Vertical Flip

a photo of a wedding dress

a photo of an old woman

View: Vertical Flip

an oil painting of albert einstein

an oil painting of elvis

View: Vertical Flip

an oil painting of a red panda

an oil painting of a teddy bear

View: Vertical Flip

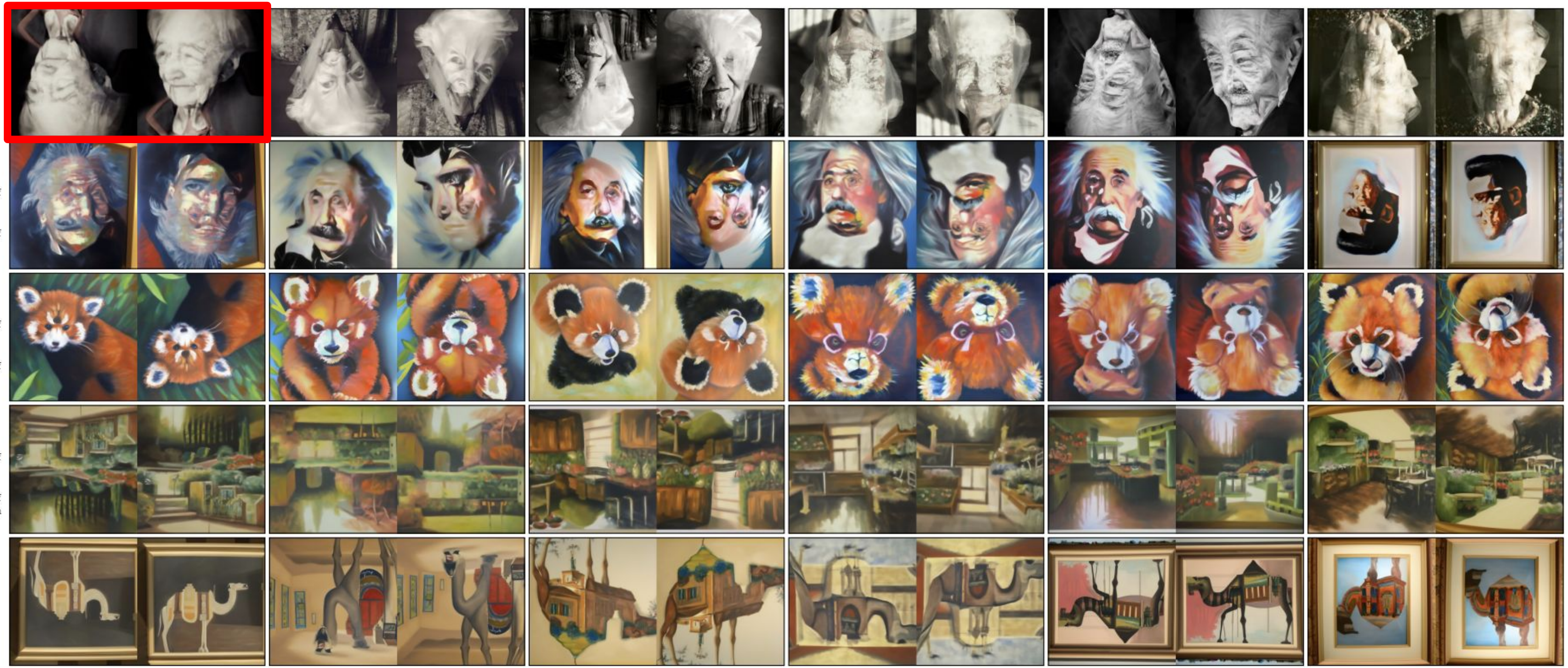
an oil painting of a kitchen

an oil painting of a botanical garden

View: Vertical Flip

a painting of a museum

a painting of a camel

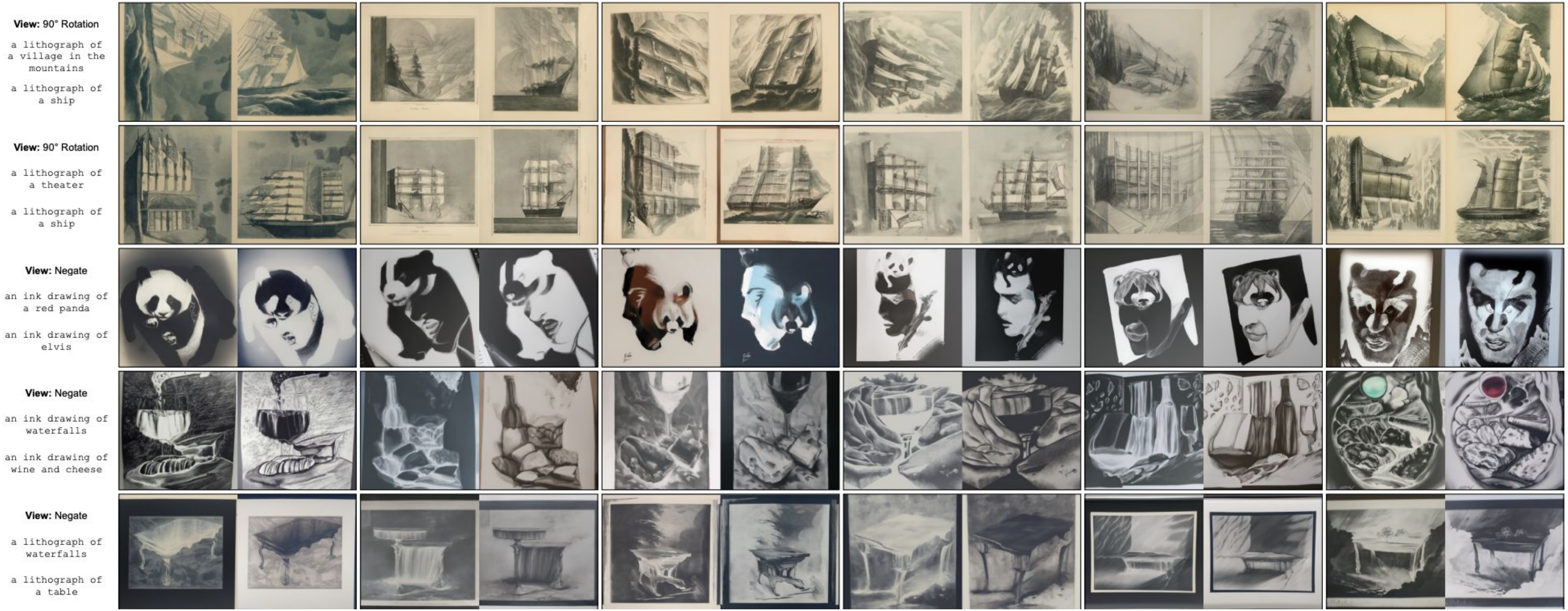


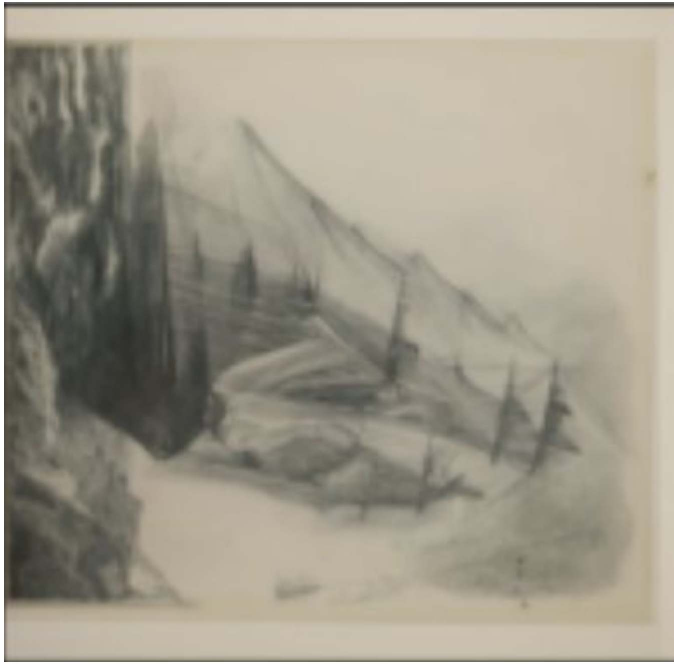
Wedding dress

Old woman

Albert Einstein

Elvis Presley


View: 90° Rotation
a lithograph of a village in the mountains
a lithograph of a ship
View: 90° Rotation
a lithograph of a theater
a lithograph of a ship
View: Negate
an ink drawing of a red panda
an ink drawing of elvis
View: Negate
an ink drawing of waterfalls
an ink drawing of wine and cheese
View: Negate
a lithograph of waterfalls
a lithograph of a table



Village in the mountains

A ship

Ink drawing of waterfalls

−

Ink drawing of wine and cheese

Key Takeaways

- Diffusion is flexible!
  - Manipulating noise can still result in coherent outputs

- Conditioning isn't the only way to incorporate info
  - This problem frequently is approached by blending prompt embeddings
  - "Conjoined" diffusion processes are a conceptually simpler way to do this!
  - **Blend noise instead!!**

What properties do we need to apply this to other domains?

- a transformation that preserves diffusion properties…
  - (in the image case, a linear and orthogonal transformation matrix)
- and an obvious place to separate the task into separate diffusion processes
  - (in the image case, diffusing two images and averaging the noise)

We can apply this to graphs!

At each step, a graph neural network:

- Aggregates information from each node’s neighborhood by averaging
  - (in a graph attention network, a weighted average of neighbors, given by softmax of attention scores)
- Transforms aggregated info with a neural network
- Replaces each node’s state with the transformed, aggregated info

We can apply this to graphs! (modified for diffusion)

At each step, a graph neural network:

(each estimate has mean of 0)

- Diffuse over each node's state
- Aggregates noise estimates from each node's neighborhood by averaging
  - (in a graph attention network, a weighted average of neighbors, given by softmax of attention scores)
  - divide softmax'd attention scores by L2 norm to preserve variance of 1
- ~~Transforms aggregated info with a neural network~~
- Replaces each node's state with the transformed, aggregated info

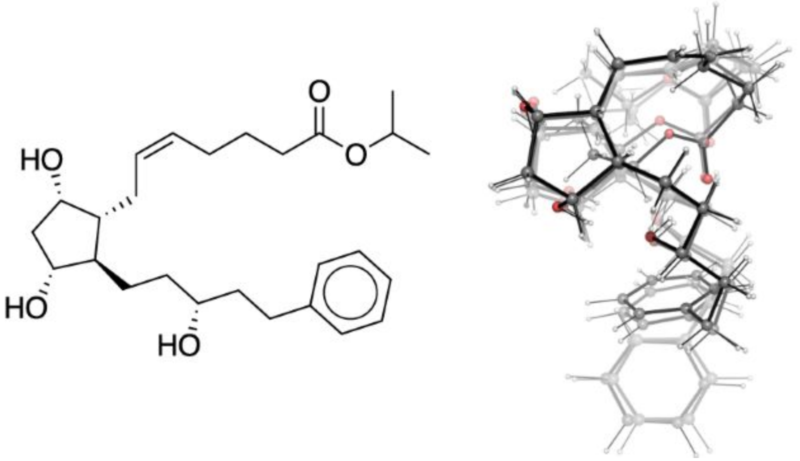
preserves diffusion properties… ✅

separate diffusion processes… ✅

Problem Definition - Molecular Conformation

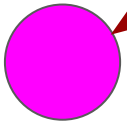
- Given a molecule graph, return a stable configuration of 3D coordinates for each atom
- Conditional diffusion: conditioned in molecule graph, diffuse coordinates

CC(C)OC(=O)CCC/C=C\\C[C@H]1[C@@H](O)C[C@@H](O)[C@@H]1CC[C@@H](O)CCc1ccccc1



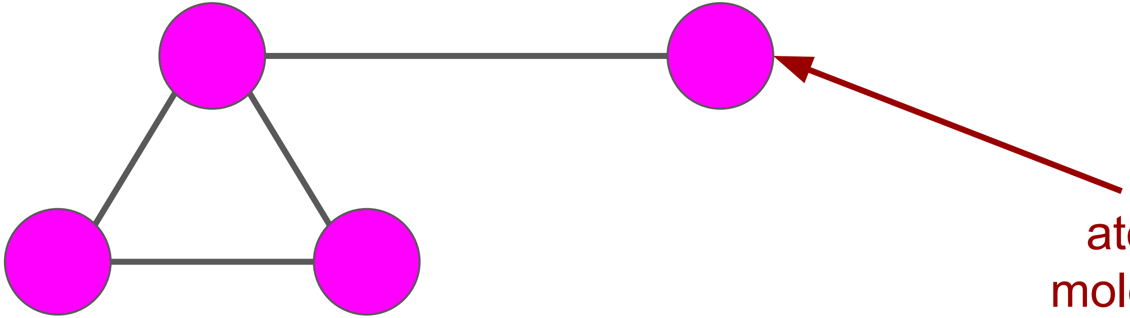
What data do we use?

an atom

 ≡ { x, y, z, atom properties, molecule properties }

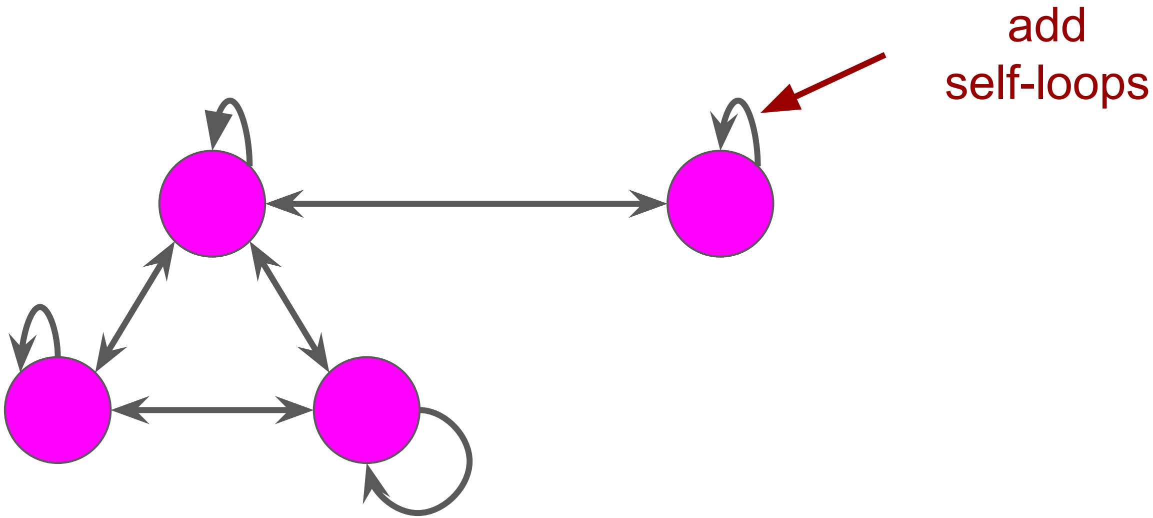
x, y, z	atom properties	molecule properties
coordinates	per atom, per molecule	same for all atoms per molecule
from qm9	from MPNN for quantum chemistry	from qm9

An example molecule:

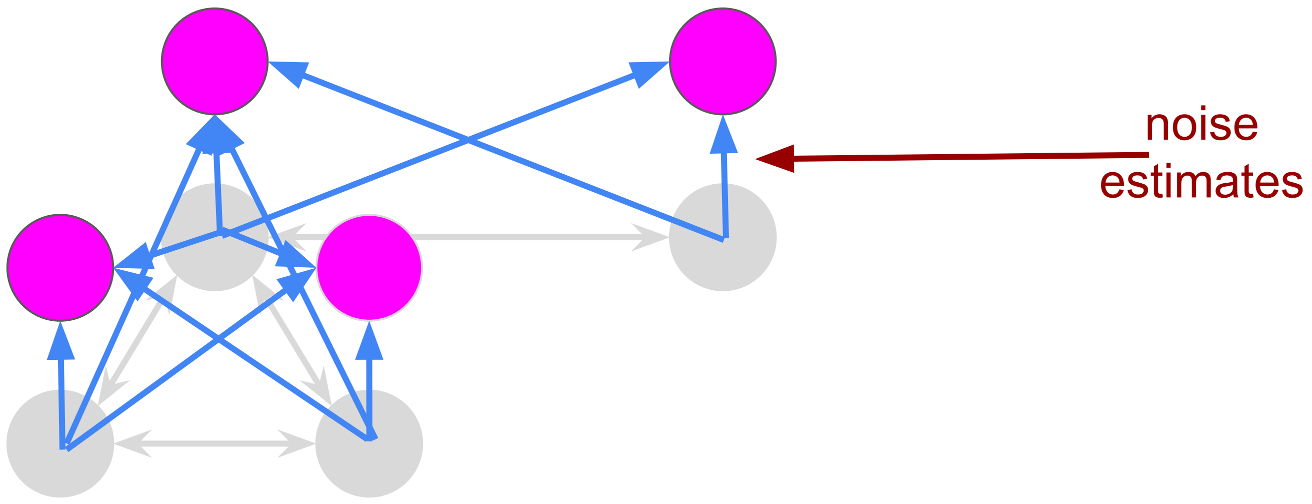


coordinates,
atom properties,
molecule properties

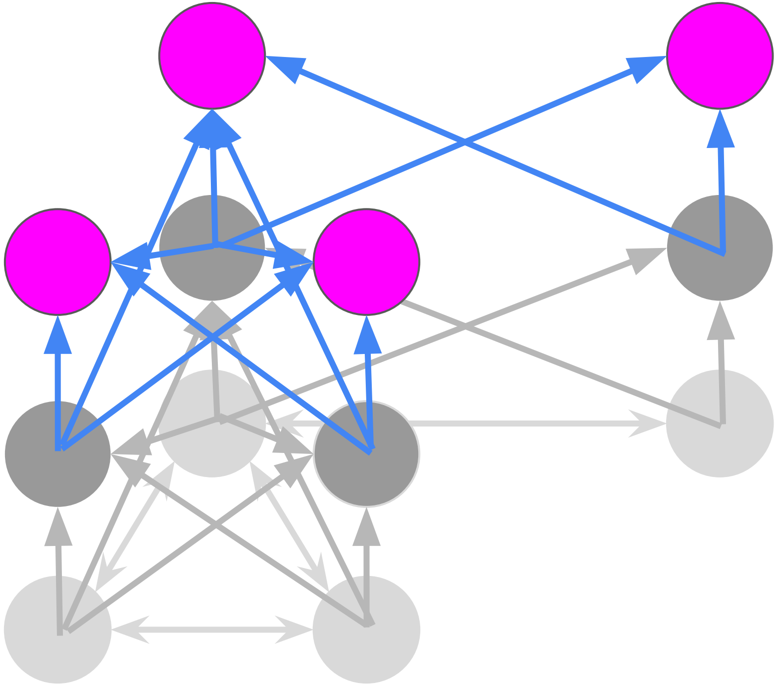
An example molecule: (at diffusion step 1)



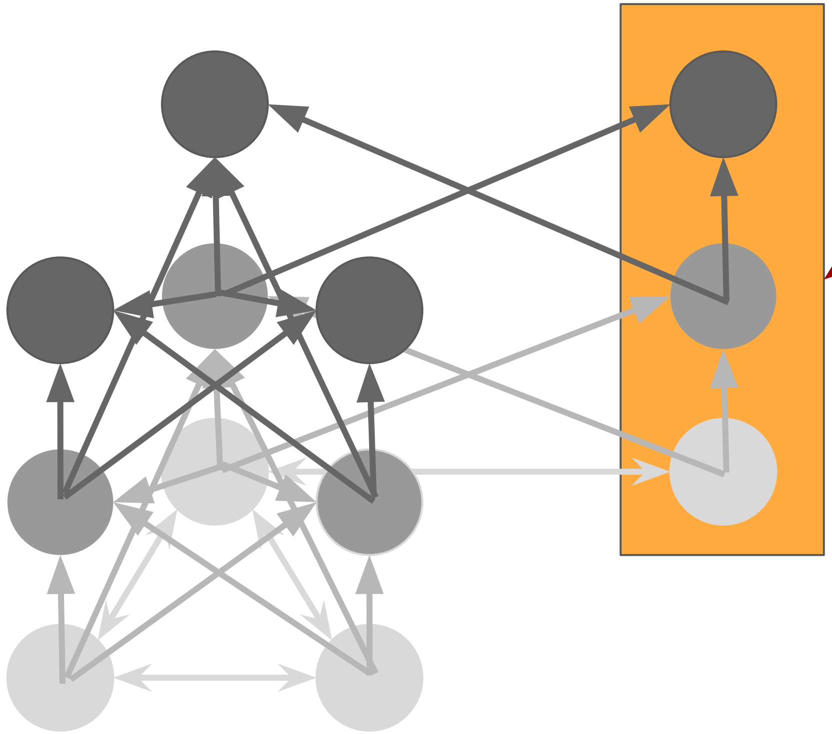
An example molecule: (at diffusion step 2)



An example molecule: (at diffusion step 3)



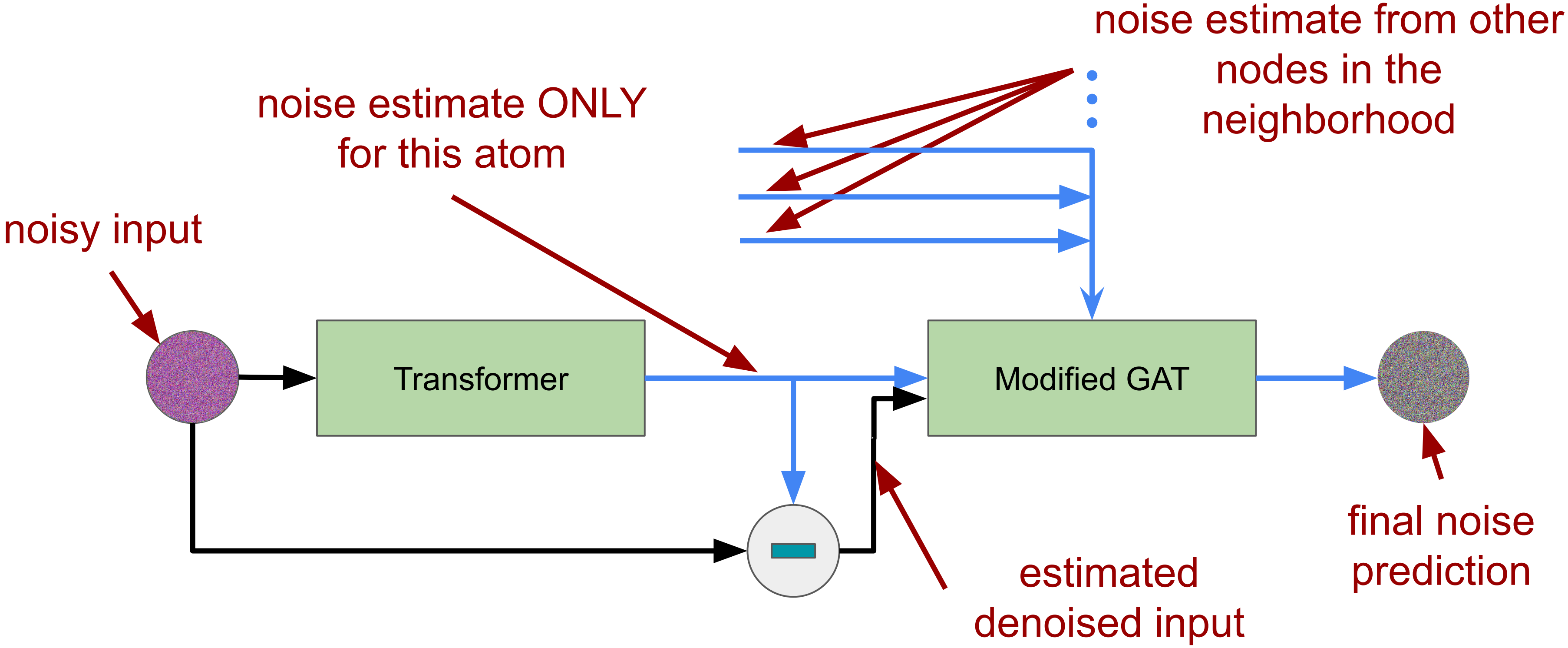
An example molecule: explained



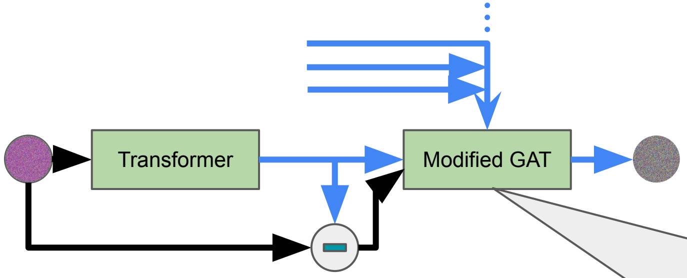
diffusion process runs on each node!

(separate diffusion processes ✅)

At each timestep, we…



At each timestep, we…



- compute attention scores for each neighbor
- normalize attention scores s.t. sum((att_x)^2 for x in neighborhood) = 1
- output = sum(att_x (noise_est) for x ….)

preserve variance!

preserve mean!

Diffusion…

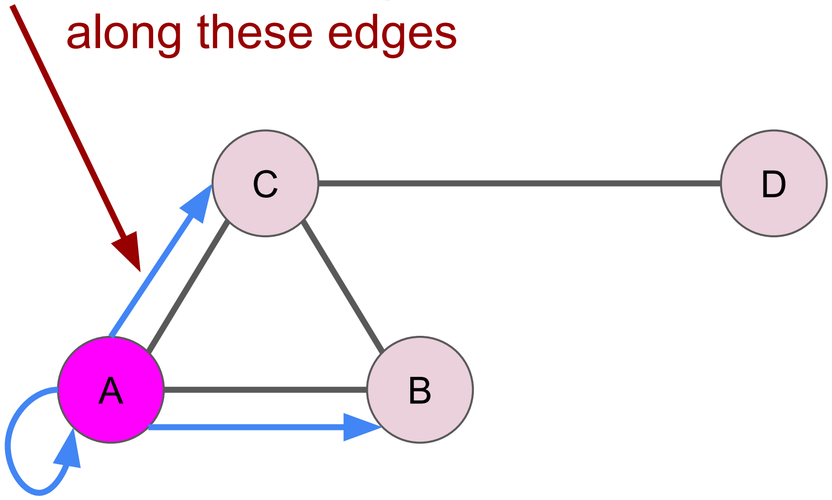
- A diffusion model diffuses through time
    - Each model step moves closer to the denoised signal
- A graph neural network propagates its information through time
    - Each GNN pass spreads information farther through the graph
- Each diffusion step only simulates a single timestep, but the method (diffusion) already takes that into account…

How do we allow the information to propagate through time,
while still only sampling individual timesteps?

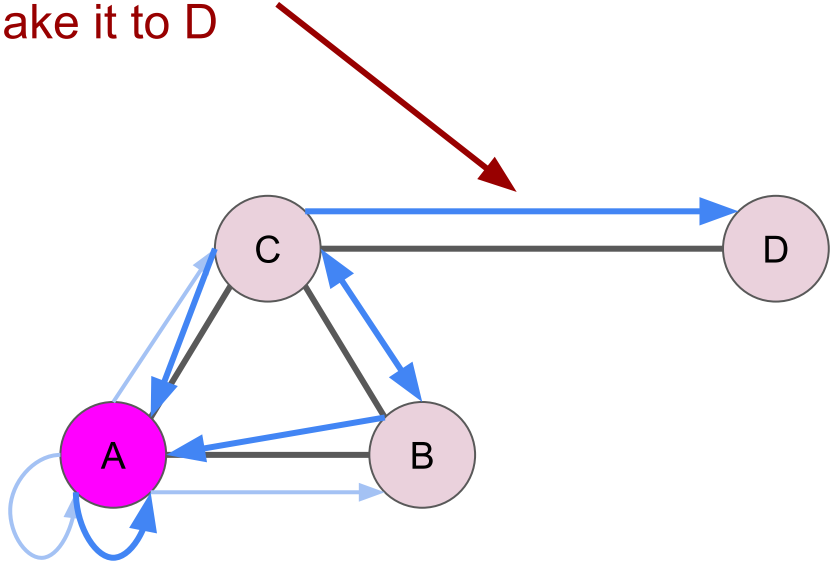
Walks on a graph

Information ONLY passes along these edges

Looking only at one step, information from A will never make it to D



walks of length 1 starting at A



walks of length 2 starting at A

Adjacency matrix trick…

- At timestep *max_timesteps* - 1, information can only propagate to adjacent nodes
- At timestep 1, information will eventually propagate everywhere
- Lets simulate this propagation in one step:
  - At timestep t out of n max_timesteps, there are $n - t$ timesteps left

All walks of length ($n - t$) = adjacency_matrix$^{(n-t)}$

this results in HUGE numbers, so we normalize adj a la GCN,
$(\text{deg}^{-½}\ \text{adj}\ \text{deg}^{-½})^{(n-t)}$

All tied together now!

simpler, respects graph structure more

- train a diffusion process that fundamentally operates on individual atoms
- diffuse directly over the properties we are interested in
- blend the noise estimates together at each diffusion step
- preserve diffusion properties by normalizing attention coefficients and averaging 0-mean noise estimates
- simulate information propagation through time with matrix exponentiation

learns conformations by learning other useful properties

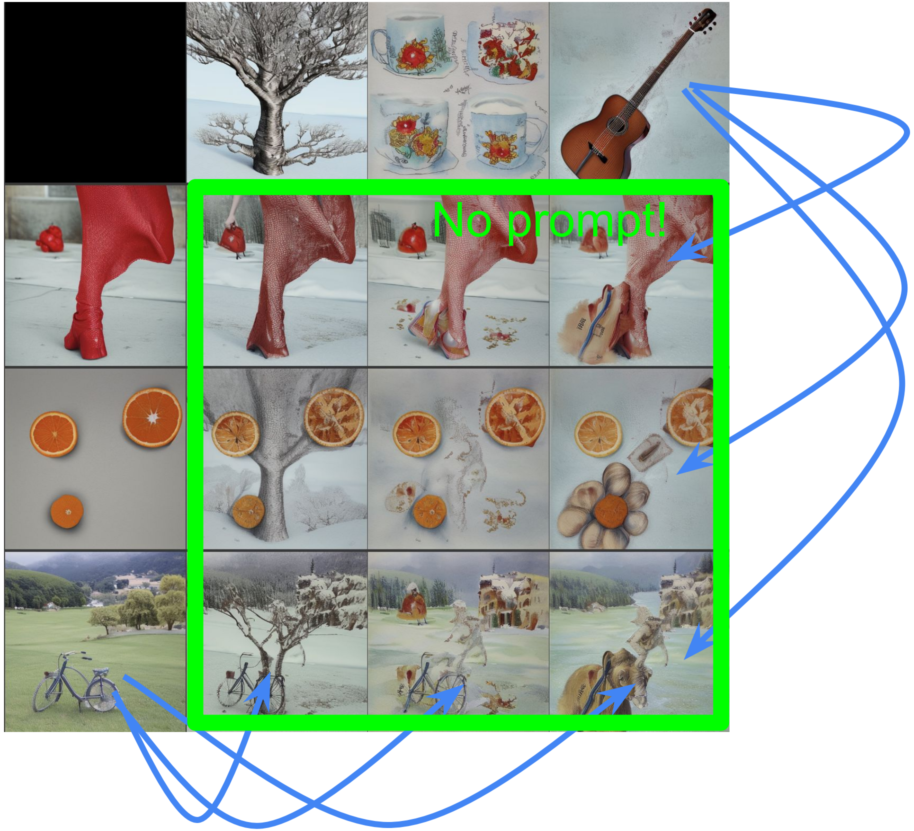
efficient!

simple correctness!

morally nice! blending noise estimates ≅ every atom exerting a “force” on its neighbors, along bonds

https://github.com/ashwinbaluja/PerNodeDiffusion
https://colab.research.google.com/drive/1d_V2bsVZdtBwpOHHOt_WowH4U8Uu7GZn?usp=sharing

Visual Anagrams ++ (latent-space) (my fun work) (extra)



Visual Anagrams ++ (pixel-space) (my fun work) (extra)