Deep Learning A-Z

Jisang Han

onground@korea.ac.kr

KUGODS

Department of Computer Science and Engineering, Korea University

Korea University GOogle Developer Students
KUGODS

Ice Breaking

- **Why?**
- **What I've studied**
- **What I want to study**
- **What I expect from the study?**

Ice Breaking

밥 먹으러 갈 사람?

Plans

- **Deep Learning for Projects**
  - PyTorch
  - Practice importing and using someone else's code
  - Make your own Toy Project

- **Competition**
  - KUGODS Kaggle Competition
  - Computer Vision

Deep Learning Tasks

- **TI2V**

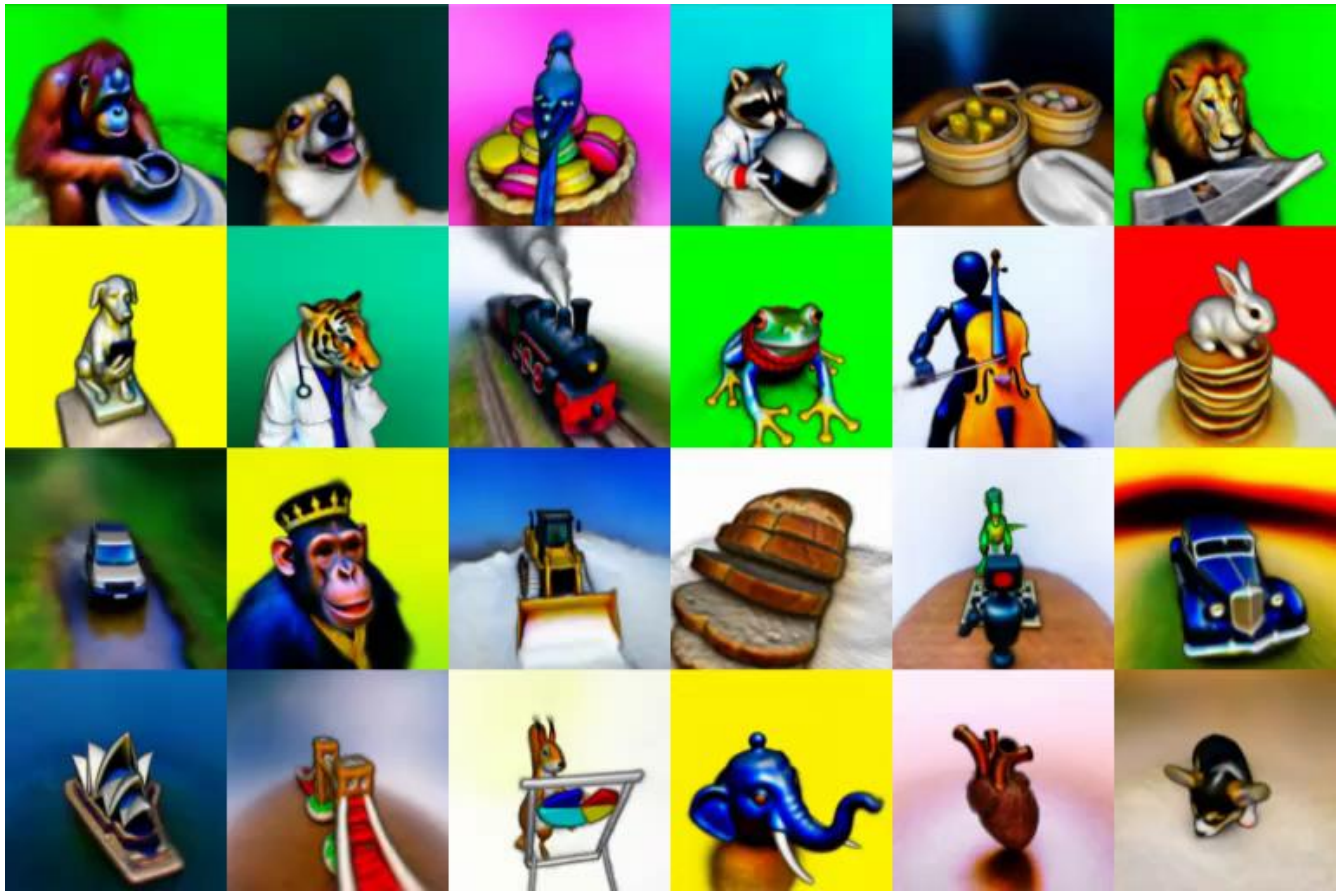
https://makeavideo.studio/

A dog wearing a Superhero outfit with red cape flying through the sky

Deep Learning Tasks

- **Text-To-3D**

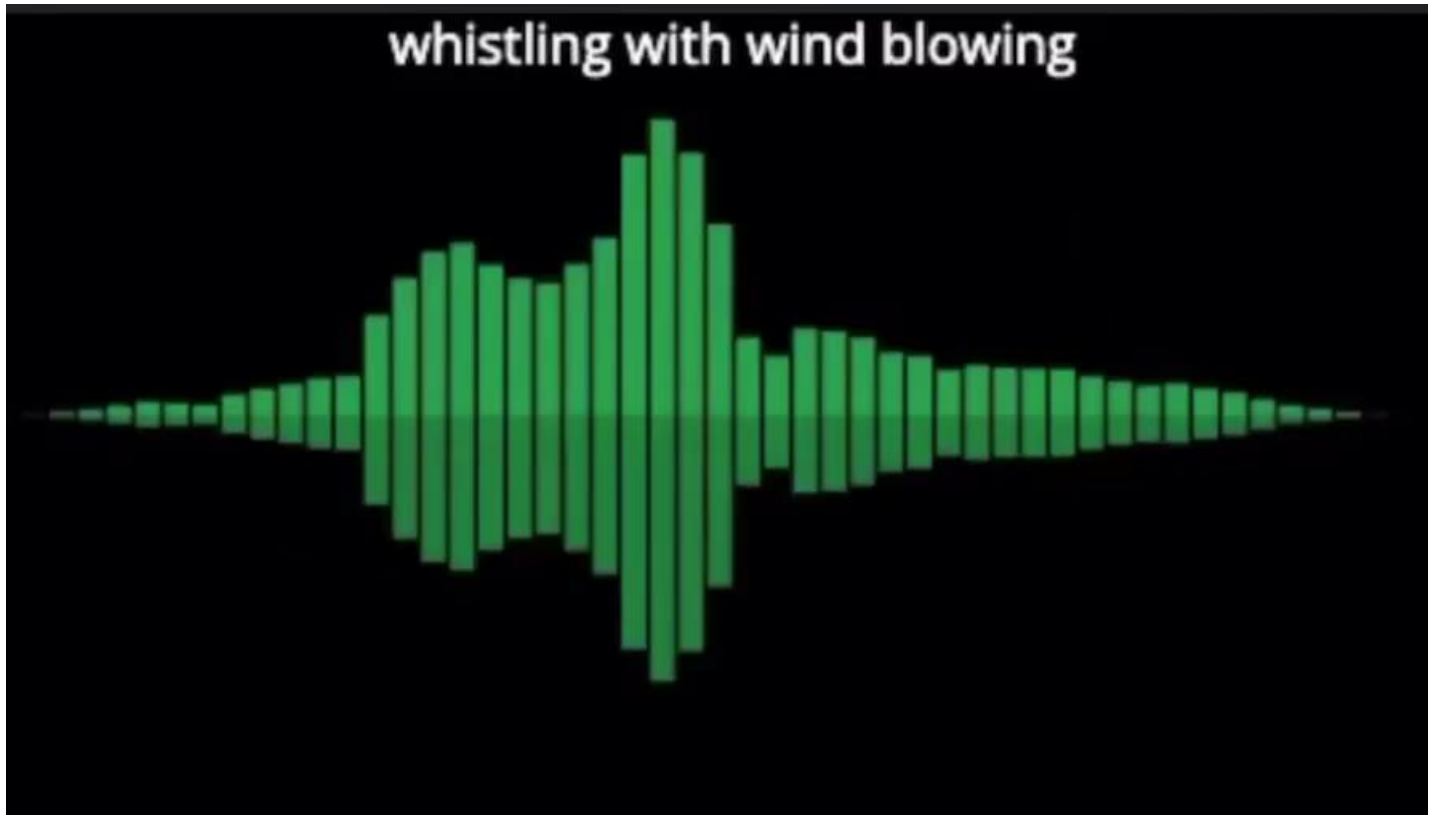
https://dreamfusion3d.github.io/



Deep Learning Tasks

- **Text-To-Audio**

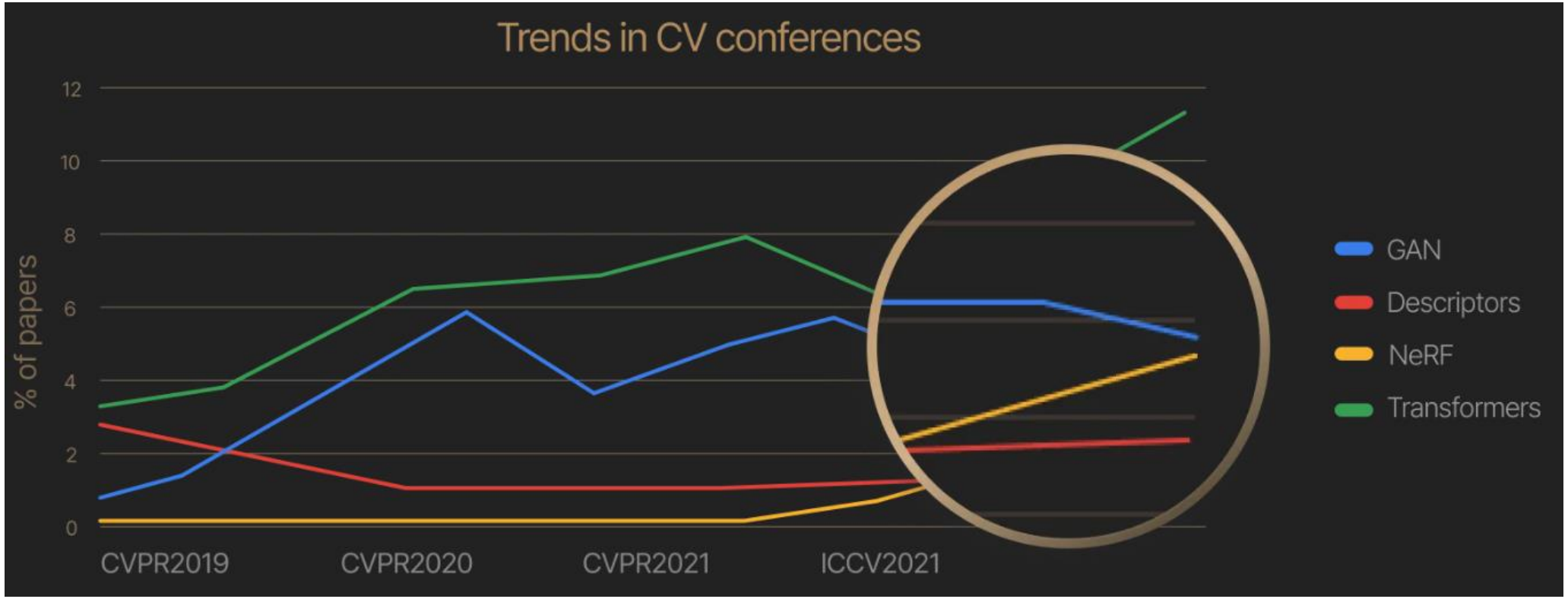
https://felixkreuk.github.io/text2audio_arxiv_samples/



Deep Learning Tasks

- **NeRF**

https://www.matthewtancik.com/nerf



Deep Learning Pipeline

- Model
- Datasets
- Train / Validation
- Prediction / Testing

Deep Learning Pipeline

- Model

```python
class CNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer = nn.Sequential(
            nn.Conv2d(3, 16, 3, padding=1),
            nn.BatchNorm2d(16),
            nn.ReLU(),

            nn.Conv2d(16, 32, 3, padding=1),
            nn.BatchNorm2d(32),
            nn.ReLU(),

            nn.MaxPool2d(2, 2),

            nn.Conv2d(32, 64, 3, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),

            nn.MaxPool2d(2, 2),
        )

        self.fc_layer = nn.Sequential(
            nn.Linear(200704, 128),
            nn.BatchNorm1d(128),
            nn.ReLU(),

            nn.Linear(128, 64),
            nn.BatchNorm1d(64),
            nn.ReLU(),

            nn.Linear(64, 2),
        )

    def forward(self, x):
        out = self.layer(x)
        out = out.view(out.size(0), -1)
        out = self.fc_layer(out)
        return out

model = CNN().to('cuda')
```

Deep Learning Pipeline

- Datasets

```python
class dataset(torch.utils.data.Dataset):
    def __init__(self, file_list, transform=None):
        self.file_list = file_list
        self.transform = transform

    #dataset length
    def __len__(self):
        self.filelength = len(self.file_list)
        return self.filelength

    #load an one of images
    def __getitem__(self, idx):
        img_path = self.file_list[idx]
        img = Image.open(img_path)
        img_transformed = self.transform(img)

        label = img_path.split('/')[-1].split('.')[0]
        if label == 'dog':
            label = 1
        elif label == 'cat':
            label = 0

        return img_transformed, label


train_data = dataset(train_list, transform=data_transforms)
train_loader = torch.utils.data.DataLoader(dataset=train_data, batch_size=batch_size, shuffle=True)
```

Deep Learning Pipeline

- Train / Validation

```python
learning_rate = 1e-3
batch_size = 64

def train_loop(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    for batch, (X, y) in enumerate(dataloader):
        # 예측(prediction)과 손실(loss) 계산
        X = X.to('cuda')
        y = y.to('cuda')
        pred = model(X)
        loss = loss_fn(pred, y)

        # 역전파
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch % 10 == 0:
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")


def test_loop(dataloader, model, loss_fn):
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    test_loss, correct = 0, 0

    with torch.no_grad():
        for X, y in dataloader:
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()

    test_loss /= num_batches
    correct /= size
    print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {test_loss:>8f} \n")

loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

epochs = 10
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train_loop(train_loader, model, loss_fn, optimizer)
print("Done!")
```

Deep Learning Pipeline

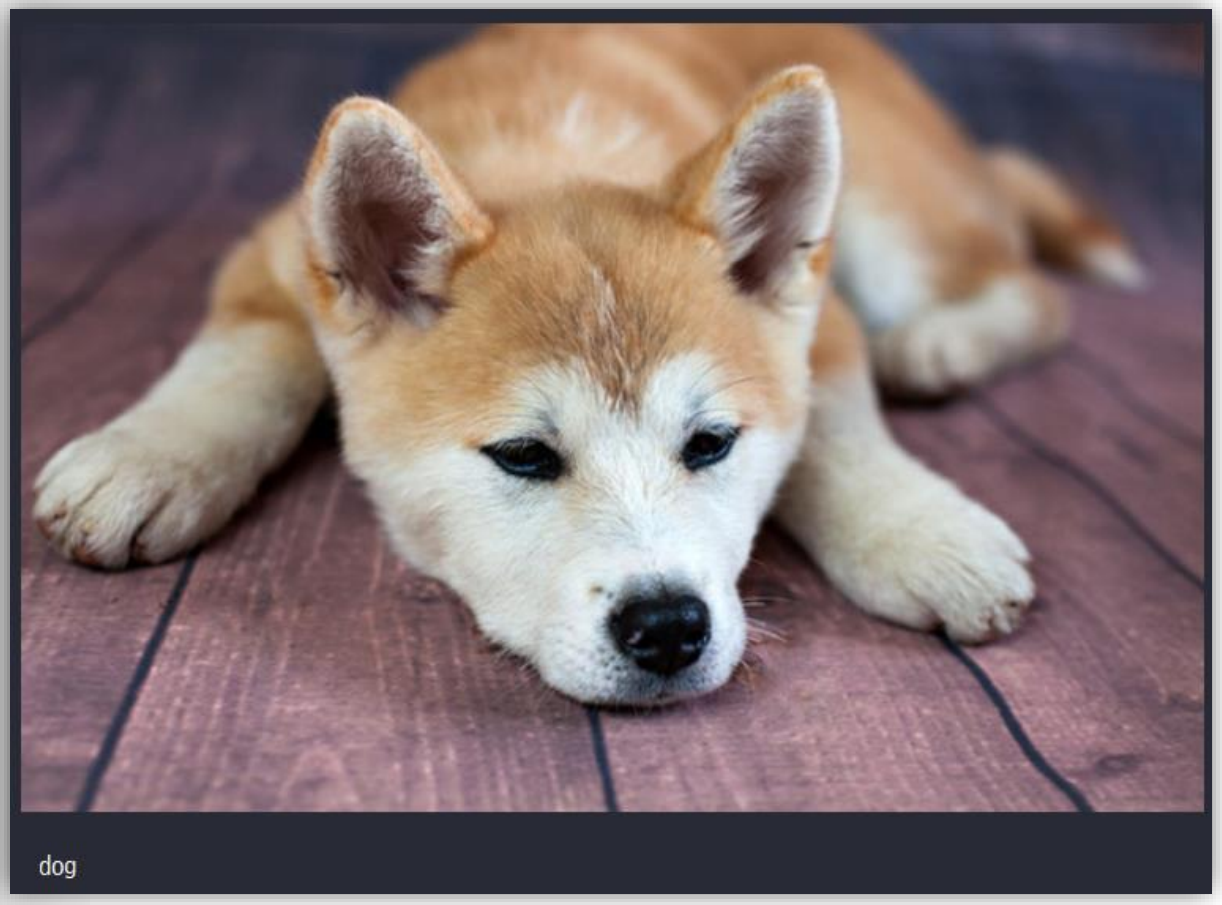
- Prediction

```
#test
from PIL import Image
import numpy as np
from IPython.display import display
%matplotlib inline

model = torch.load('catdog.pth')
model.eval()

def test(path):
    img = Image.open(path)
    x = data_transforms(img).unsqueeze(0).to('cuda')
    y = model(x)
    display(img)
    if y.argmax(1) == 1:
        print('dog')
    elif y.argmax(1) == 0:
        print('cat')

test('54607_56591_5215.jpg')
```



Thank you!
Q & A