DATA SCIENCE BRAIN
@datasciencebrain

GANS

CHEAT SHEET

Using Tensorflow

Save for later reference ·····→

01 GAN ARCHITECTURE

```
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, LeakyReLU,
BatchNormalization, Reshape, Flatten, Input

# Generator model
generator = Sequential([
    Dense(256, input_dim=noise_dim),
    LeakyReLU(alpha=0.2),
    BatchNormalization(),
    Dense(512),
    LeakyReLU(alpha=0.2),
    BatchNormalization(),
    Dense(1024),
    LeakyReLU(alpha=0.2),
    BatchNormalization(),
    Dense(output_dim, activation='tanh')
])
```

01 GAN ARCHITECTURE

```
# Discriminator model
discriminator = Sequential([
 Dense(1024, input_dim=output_dim),
 LeakyReLU(alpha=0.2),
 Dense(512),
 LeakyReLU(alpha=0.2),
 Dense(256),
 LeakyReLU(alpha=0.2),
 Dense(1, activation='sigmoid')
])

# Combined model (GAN)
discriminator.trainable = False
gan_input = Input(shape=(noise_dim,))
x = generator(gan_input)
gan_output = discriminator(x)
gan = Model(gan_input, gan_output)
```

02 GAN TRAINING

```
# Compile the discriminator
discriminator.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Compile the GAN
gan.compile(loss='binary_crossentropy', optimizer='adam')

# Training loop
for epoch in range(epochs):
    # Generate random noise samples
    noise = np.random.normal(0, 1, size=(batch_size, noise_dim))

    # Generate fake images
    generated_images = generator.predict(noise)

    # Get real images from the dataset
    real_images = get_real_images_from_dataset()

    # Labels for the discriminator training
    real_labels = np.ones((batch_size, 1))
    fake_labels = np.zeros((batch_size, 1))

    # Train the discriminator on real and fake images
    d_loss_real = discriminator.train_on_batch(real_images, real_labels)
    d_loss_fake = discriminator.train_on_batch(generated_images, fake_labels)

    # Combined loss for the GAN
    d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

    # Train the generator via the GAN model
    noise = np.random.normal(0, 1, size=(batch_size, noise_dim))
    valid_labels = np.ones((batch_size, 1))
    g_loss = gan.train_on_batch(noise, valid_labels)

    # Print progress
    print(f"Epoch {epoch}/{epochs} [D loss: {d_loss[0]} | D accuracy: {100 * d_loss[1]}] [G loss: {g_loss}]")
```

03 GENERATING IMAGES

```
# Generate new images using the trained generator
def generate_images(generator, noise, examples=16, dim=(4, 4), figsize=(10, 10)):
    generated_images = generator.predict(noise)
    generated_images = 0.5 * generated_images + 0.5  # Rescale images to [0, 1]

    plt.figure(figsize=figsize)
    for i in range(generated_images.shape[0]):
        plt.subplot(dim[0], dim[1], i + 1)
        plt.imshow(generated_images[i, :, :, 0], interpolation='nearest', cmap='gray_r')
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# Generate random noise for image generation
noise = np.random.normal(0, 1, size=(16, noise_dim))
generate_images(generator, noise)
```