**Отчет на задание №1**

В данном задании было предложено Реализовать ИНС (искусственную нейронную сеть) - автоэнкодер. Задача автоэнкодера восстановить вход на выходе.

В качестве датасета для обучения была выбрана база Stanford Cars Dataset <https://www.kaggle.com/datasets/jessicali9530/stanford-cars-dataset?resource=download>

На первом этапе было необходимо определиться с архитектурой нейронной сети. На старте попытки давали не очень удовлетворительный результат – картинки оставались слишком размытыми

**Примеры используемых моделей**

*x = Conv2D(128, (3, 3), activation='relu', padding='same')(input\_img)*

*x = MaxPooling2D((2, 2), padding='same')(x)*

*x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)*

*x = MaxPooling2D((2, 2), padding='same')(x)*

*x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)*

*x = MaxPooling2D((2, 2), padding='same')(x)*

*x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)*

*x = MaxPooling2D((2, 2), padding='same')(x)*

*encoded = Conv2D(1, (8, 8), activation='relu', padding='same')(x)*

*input\_encoded = Input(shape=(8, 8, 1))*

*x = Conv2D(16, (3, 3), activation='relu', padding='same')(input\_encoded)*

*x = UpSampling2D((2, 2))(x)*

*x = Conv2D(32, (2, 2), activation='relu', padding='same')(x)*

*x = UpSampling2D((2, 2))(x)*

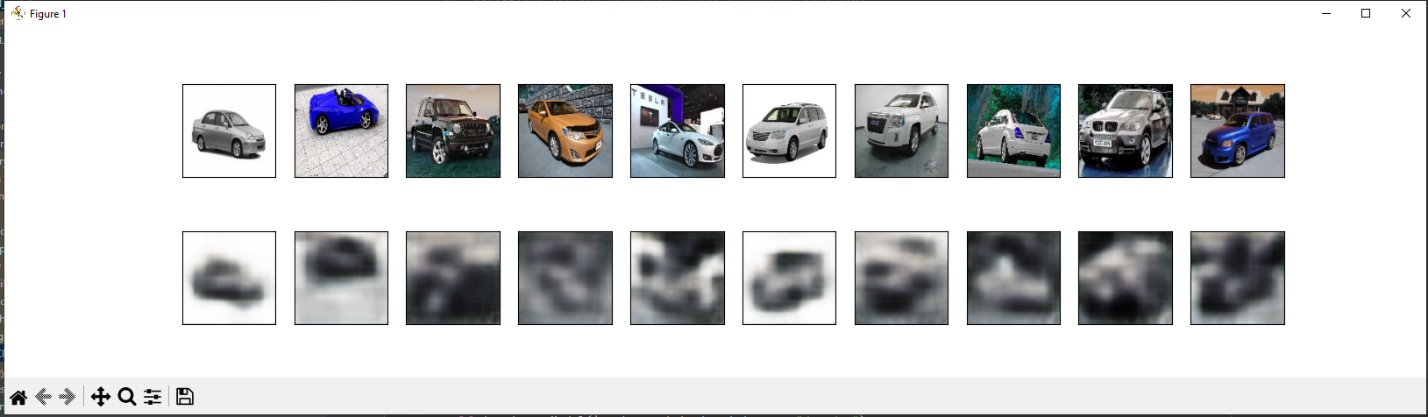
*x = Conv2D(64, (2, 2), activation='relu', padding='same')(x)*

*x = UpSampling2D((2, 2))(x)*

*x = Conv2D(128, (2, 2), activation='relu', padding='same')(x)*

*x = UpSampling2D((2, 2))(x)*

*decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)*

**

*x = Conv2D(filters=32, kernel\_size=(3, 3), activation='elu', padding='same')(x)*

*x = MaxPooling2D(pool\_size=(2, 2)(x))*

*x = Conv2D(filters=64, kernel\_size=(3, 3), activation='elu', padding='same'))*

*x = MaxPooling2D(pool\_size=(2, 2)))*

*x = Conv2D(filters=128, kernel\_size=(3, 3), activation='elu', padding='same'))*

*x = MaxPooling2D(pool\_size=(2, 2)))*

*x = Conv2D(filters=256, kernel\_size=(3, 3), activation='elu', padding='same'))*

*x = MaxPooling2D(pool\_size=(2, 2)))*

*x = Flatten())*

*x = Dense(32))*

*# decoder*

*Input = InputLayer((32,))*

*x = Dense(14 \* 14 \* 256)(input)*

*x = Reshape((14, 14, 256))(x)*

*x = Conv2DTranspose(filters=128, kernel\_size=(3, 3), strides=2, activation='elu', padding='same')(x)*

*x = Conv2DTranspose(filters=64, kernel\_size=(3, 3), strides=2, activation='elu', padding='same')(x)*

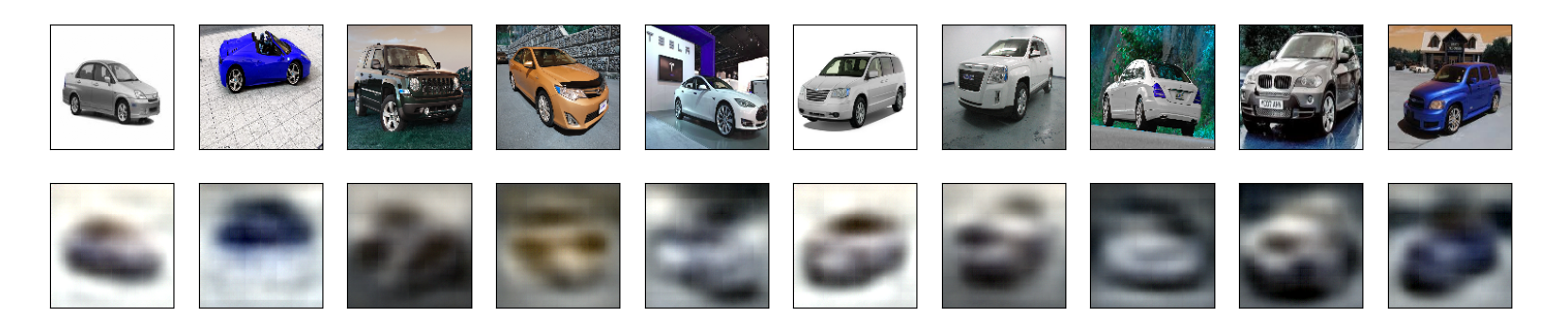
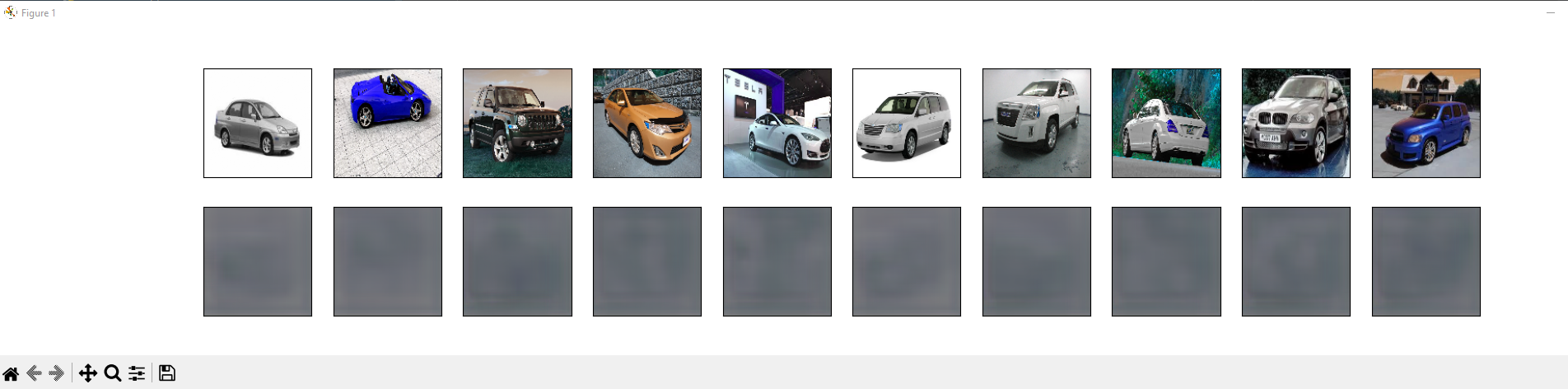
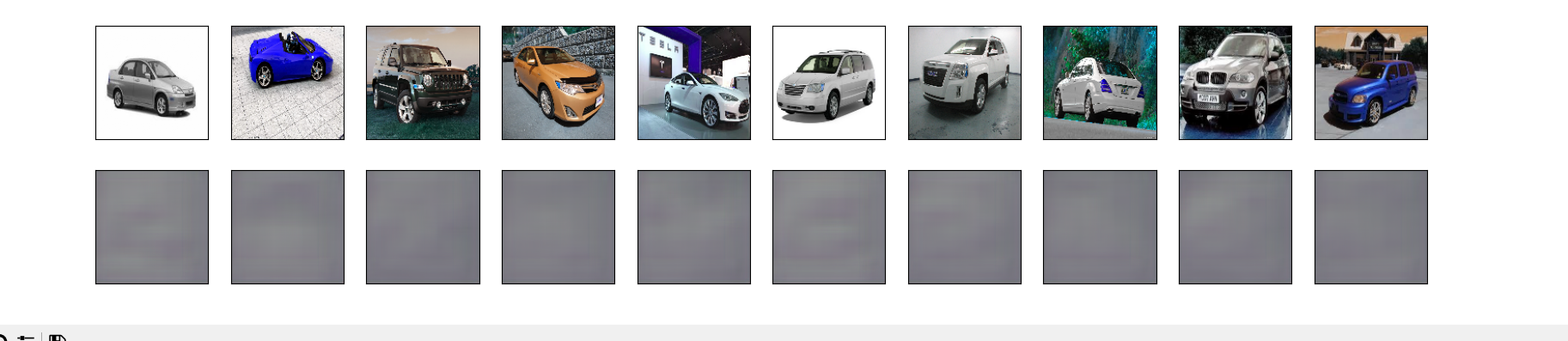
*x = Conv2DTranspose(filters=32, kernel\_size=(3, 3), strides=2, activation='elu', padding='same')(x)*

*x = Conv2DTranspose(filters=3, kernel\_size=(3, 3), strides=2, activation=None, padding='same')(x)*



Увеличение и уменьшение кол-ва слоёв не улучшило результат

**Примеры:**



После этого было испробовано отказаться от слоёв *MaxPooling2D* и производить свёртку в слое  *Conv2D* путём добавления аргумента *strides.* Так же было убрано разделение на энкодер/декодер и все слои были описаны внутри модели *autoencoder*

*input\_img = Input(shape=(224, 224, 3))*

*x = Conv2D(128, (3, 3), activation='relu', strides=2, padding='same')(input\_img)*

*x = BatchNormalization()(x)*

*x = Conv2D(256, (3, 3), activation='relu', strides=2, padding='same')(x)*

*x = BatchNormalization()(x)*

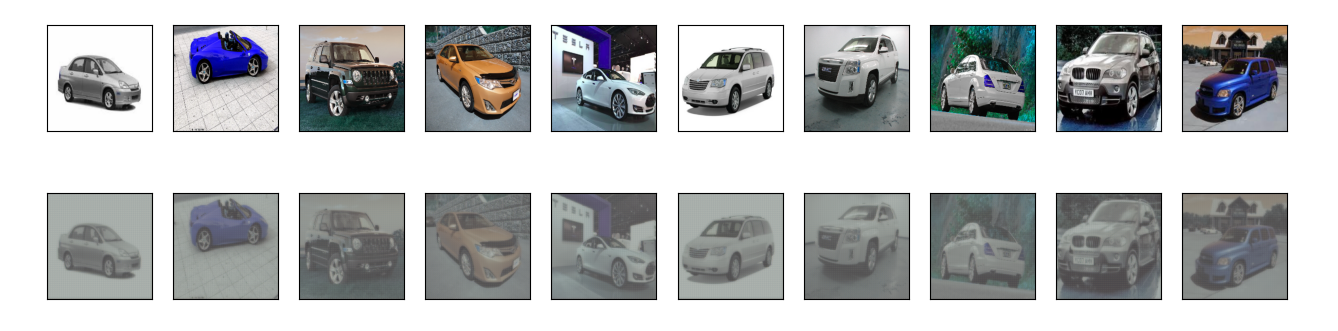
*x = Conv2DTranspose(filters=256, kernel\_size=(3, 3), strides=2, activation='relu', padding='same')(x)*

*x = BatchNormalization()(x)*

*x = Conv2DTranspose(filters=128, kernel\_size=(3, 3), strides=2, activation='relu', padding='same')(x)*

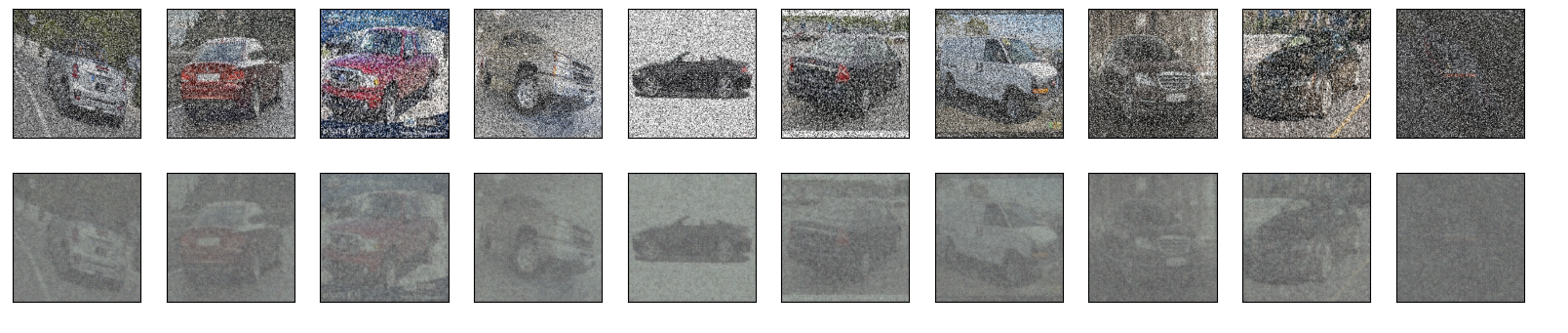
*x = BatchNormalization()(x)*

*decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)*



Результат стал намного лучше. Можно работать.

Добавляем шум на изображение



Как видно, автоэнкодер пытается воспроизвести изображение в изначальном виде, т.е. с шумами

Попробуем обучать модель следующим образом: На вход модели подаём обработанное (зашумленное) изображение, но в качестве результата подаём изображение без шума. Т.е. будем ожидать, что автоэнкодер научится избавляться от фильтра соль/перец



Похоже, что работает. Увеличиваем кол-во слоёв, увеличиваем обучающую выборку до 7000 и кол-во эпох до 20. Размер входного изображения 224\*224. Добавляем слои BatchNirmalization

input\_img = Input(shape=(self.\_\_img\_shape, self.\_\_img\_shape, 3))  
x = Conv2D(64, (3, 3), activation='relu', strides=2, padding='same')(input\_img)  
x = BatchNormalization()(x)  
x = Conv2D(128, (3, 3), activation='relu', strides=2, padding='same')(x)  
x = BatchNormalization()(x)  
x = Conv2D(256, (3, 3), activation='relu', strides=2, padding='same')(x)  
x = BatchNormalization()(x)  
x = Conv2D(512, (3, 3), activation='relu', strides=2, padding='same')(x)  
x = BatchNormalization()(x)  
x = Conv2DTranspose(filters=512, kernel\_size=(3, 3), strides=2, activation='relu', padding='same')(x)  
x = BatchNormalization()(x)  
x = Conv2DTranspose(filters=256, kernel\_size=(3, 3), strides=2, activation='relu', padding='same')(x)  
x = BatchNormalization()(x)  
x = Conv2DTranspose(filters=128, kernel\_size=(3, 3), strides=2, activation='relu', padding='same')(x)  
x = BatchNormalization()(x)  
x = Conv2DTranspose(filters=64, kernel\_size=(3, 3), strides=2, activation='relu', padding='same')(x)  
x = BatchNormalization()(x)  
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)

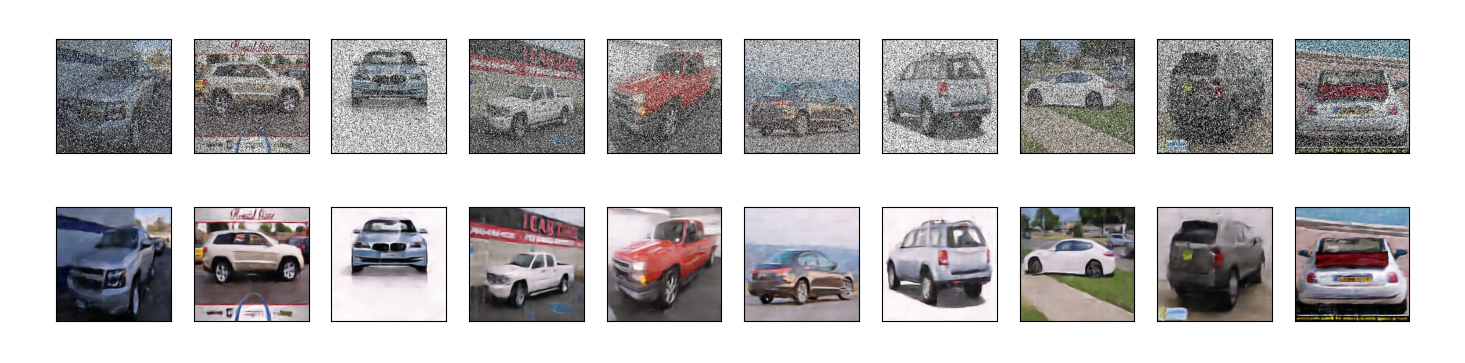
Summary выглядит следующим образом

*Total params: 5,468,931*

*Trainable params: 5,465,091*

*Non-trainable params: 3,840*

Получаем результат



Считаю его вполне приемлемым