In [13]:

```
# Лабораторная работа №5. Применение сверточных нейронных сетей (бинарная классифик

# TensorFlow и tf.keras
import tensorflow as tf
from tensorflow import keras
from keras import regularizers

import numpy as np
import matplotlib.pyplot as plt
import pdb
import os
import scipy.io
from sklearn.model_selection import train_test_split
import tarfile
from six.moves import cPickle as pickle
import zipfile
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMAGE_SIZE = 150
```

Задание 1. Загрузите данные. Разделите исходный набор данных на обучающую, валидационную и контрольную выборки.

In [15]:

```
# Useful links:
# 1) https://machinelearningmastery.com/how-to-load-large-datasets-from-directories

def extract_dataset(name):
    zip_path = os.path.join('../data/cats_vs_dogs/', name + '.zip')
    if not os.path.exists(os.path.join('../data/cats_vs_dogs/', name)):
        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall("../data/cats_vs_dogs")

extract_dataset('train')

def get_file_list_from_dir(folder_path):
    all_files = os.listdir(folder_path)
    data_files = list(filter(lambda file: file.endswith('.jpg'), all_files))
    return data_files


filelist = get_file_list_from_dir('../data/cats_vs_dogs/train')
```

In [16]:

```
train_folder = '../data/cats_vs_dogs/train'
valid_folder = '../data/cats_vs_dogs/valid'
test_folder = '../data/cats_vs_dogs/test'

train_cats_len = len(os.listdir(os.path.join(train_folder, 'cats') ))
train_dogs_len = len(os.listdir(os.path.join(train_folder, 'dogs')))
valid_cats_len = len(os.listdir(os.path.join(valid_folder, 'cats') ))
valid_dogs_len = len(os.listdir(os.path.join(valid_folder, 'dogs') ))
test_cats_len = len(os.listdir(os.path.join(test_folder, 'cats')))
test_dogs_len = len(os.listdir(os.path.join(test_folder, 'dogs')))

print(train_cats_len, train_dogs_len, valid_cats_len, valid_dogs_len, test_cats_len

# using rescale because of empty images issue (https://stackoverflow.com/questions/
data_generator = ImageDataGenerator(rescale=1./255)

train_iterator = data_generator.flow_from_directory(train_folder, class_mode='binar
valid_iterator = data_generator.flow_from_directory(valid_folder, class_mode='binar
test_iterator = data_generator.flow_from_directory(test_folder, class_mode='binary'
```

```
8725 8775 1276 1225 2499 2500
Found 17500 images belonging to 2 classes.
Found 2501 images belonging to 2 classes.
Found 4999 images belonging to 2 classes.
```

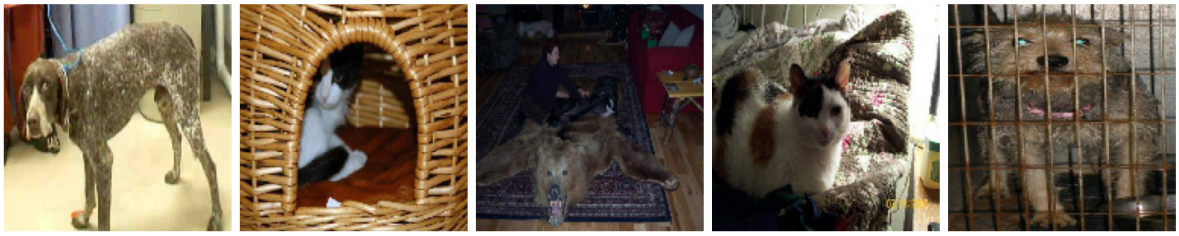
In [35]:

```
def plot_samples(iterator, sample_size, name):
    figure, axes = plt.subplots(1, 5, figsize=(20, 20))
    figure.suptitle(name)
    axes = axes.flatten()
    images = next(iterator)[0][:sample_size]
    for img, ax in zip(images, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()

plot_samples(train_iterator, 5, 'train')
plot_samples(valid_iterator, 5, 'test')
plot_samples(test_iterator, 5, 'valid')
```

train

test



valid

Задание 2. Реализуйте глубокую нейронную сеть с как минимум тремя сверточными слоями. Какое качество классификации получено?

In [30]:

```
basic_model = keras.Sequential([
    keras.layers.Conv2D(32, 3, activation='relu', input_shape=(IMAGE_SIZE, IMAGE_SI
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.1),

    keras.layers.Conv2D(64, 3, activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.1),

    keras.layers.Conv2D(128, 3, activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.1),

    keras.layers.Conv2D(256, 3, activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.1),

    keras.layers.Flatten(),

    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dropout(0.1),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
```

In [32]:

```
basic_model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accu
basic_model.summary()
basic_model_history = basic_model.fit_generator(train_iterator, steps_per_epoch=150
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
dropout_13 (Dropout)         (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
dropout_14 (Dropout)         (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
dropout_15 (Dropout)         (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 15, 15, 256)       295168
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 7, 7, 256)         0
_________________________________________________________________
dropout_16 (Dropout)         (None, 7, 7, 256)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_12 (Dense)             (None, 256)               3211520
_________________________________________________________________
dropout_17 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_13 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 257
=================================================================
Total params: 3,665,985
Trainable params: 3,665,985
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 150 steps, validate for 50 steps
Epoch 1/10
150/150 [==============================] - 191s 1s/step - loss: 0.7295
- accuracy: 0.5240 - val_loss: 0.6827 - val_accuracy: 0.5669
```

```
Epoch 2/10
150/150 [==============================] - 185s 1s/step - loss: 0.6787
- accuracy: 0.6005 - val_loss: 0.6318 - val_accuracy: 0.6356
Epoch 3/10
150/150 [==============================] - 165s 1s/step - loss: 0.6257
- accuracy: 0.6583 - val_loss: 0.5771 - val_accuracy: 0.7013
Epoch 4/10
150/150 [==============================] - 171s 1s/step - loss: 0.5780
- accuracy: 0.7008 - val_loss: 0.5793 - val_accuracy: 0.6963
Epoch 5/10
150/150 [==============================] - 153s 1s/step - loss: 0.5555
- accuracy: 0.7208 - val_loss: 0.6463 - val_accuracy: 0.6875
Epoch 6/10
150/150 [==============================] - 153s 1s/step - loss: 0.5356
- accuracy: 0.7333 - val_loss: 0.5549 - val_accuracy: 0.7337
Epoch 7/10
150/150 [==============================] - 173s 1s/step - loss: 0.5205
- accuracy: 0.7490 - val_loss: 0.4904 - val_accuracy: 0.7631
Epoch 8/10
150/150 [==============================] - 173s 1s/step - loss: 0.4966
- accuracy: 0.7663 - val_loss: 0.4679 - val_accuracy: 0.7681
Epoch 9/10
150/150 [==============================] - 178s 1s/step - loss: 0.4609
- accuracy: 0.7871 - val_loss: 0.4274 - val_accuracy: 0.8000
Epoch 10/10
150/150 [==============================] - 172s 1s/step - loss: 0.4437
- accuracy: 0.7917 - val_loss: 0.4423 - val_accuracy: 0.8125
```

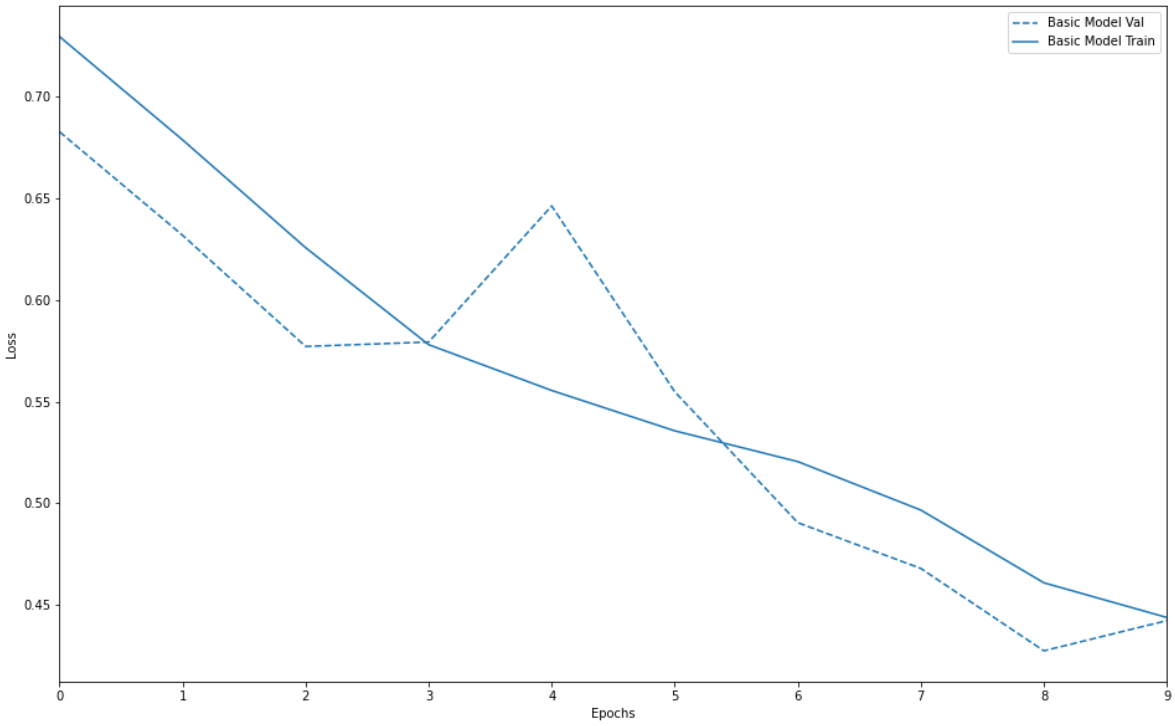
In [33]:

```
def plot_history(histories, key='binary_crossentropy'):
    plt.figure(figsize=(16,10))

    for name, history in histories:
        val = plt.plot(history.epoch, history.history['val_' + key],
                       '--', label=name.title()+' Val')
        plt.plot(history.epoch, history.history[key], color=val[0].get_color(),
                 label=name.title()+' Train')

    plt.xlabel('Epochs')
    plt.ylabel(key.replace('_',' ').title())
    plt.legend()
    plt.xlim([0,max(history.epoch)])
    plt.show()

plot_history([('basic model', basic_model_history)], key='loss')
```



Задание 3. Примените дополнение данных (data augmentation). Как это повлияло на качество классификатора?

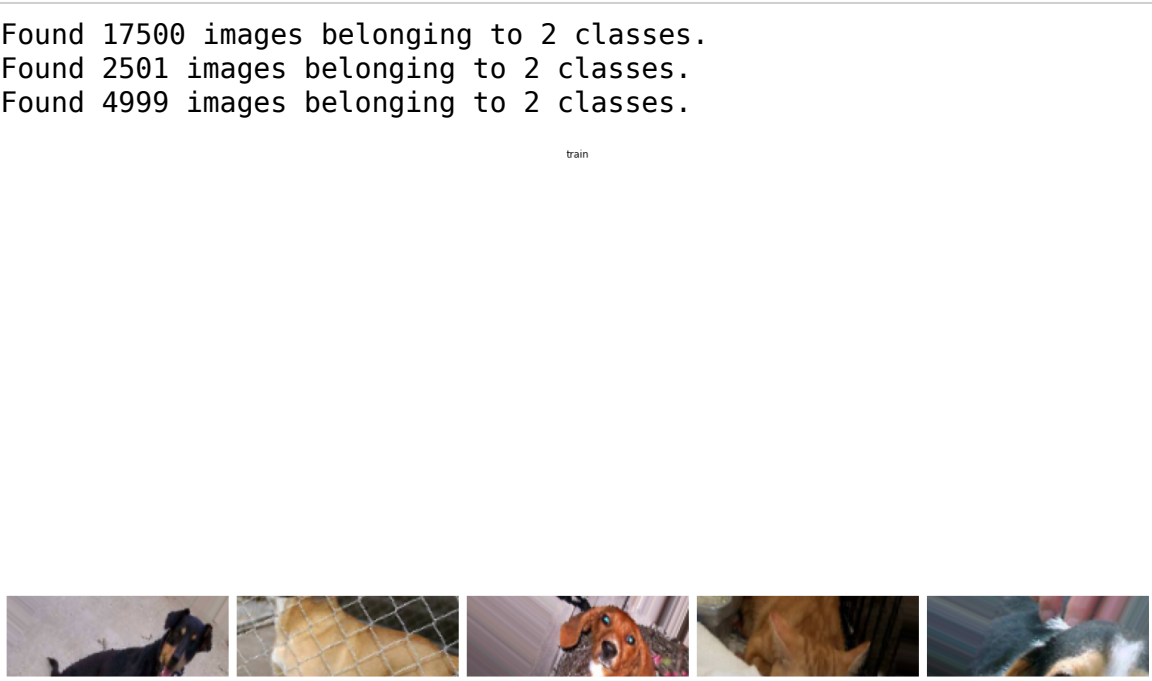
In [36]:

```
aug_data_generator = ImageDataGenerator(rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')


aug_train_iterator = aug_data_generator.flow_from_directory(train_folder, class_mod
aug_valid_iterator = aug_data_generator.flow_from_directory(valid_folder, class_mod
aug_test_iterator = aug_data_generator.flow_from_directory(test_folder, class_mode=

plot_samples(aug_train_iterator, 5, 'train')
plot_samples(aug_valid_iterator, 5, 'test')
plot_samples(aug_test_iterator, 5, 'valid')
```

```
Found 17500 images belonging to 2 classes.
Found 2501 images belonging to 2 classes.
Found 4999 images belonging to 2 classes.
```



In [37]:

```
aug_model = keras.Sequential([
    keras.layers.Conv2D(32, 3, activation='relu', input_shape=(IMAGE_SIZE, IMAGE_SI
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.1),

    keras.layers.Conv2D(64, 3, activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.1),

    keras.layers.Conv2D(128, 3, activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.1),

    keras.layers.Conv2D(256, 3, activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.1),

    keras.layers.Flatten(),

    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dropout(0.1),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
```

In [40]:

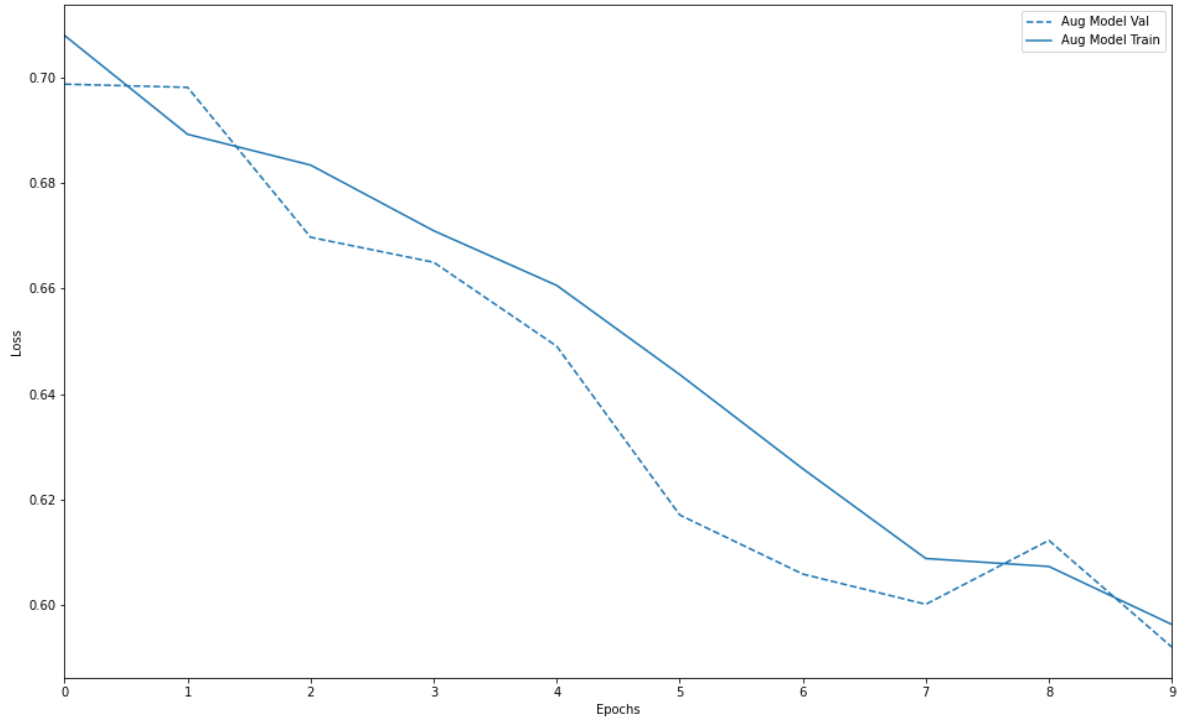
```
aug_model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accura
aug_model.summary()
aug_model_history = aug_model.fit(aug_train_iterator, steps_per_epoch=150, epochs=1
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
dropout_18 (Dropout)         (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
dropout_19 (Dropout)         (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
dropout_20 (Dropout)         (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 15, 15, 256)       295168
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 7, 7, 256)         0
_________________________________________________________________
dropout_21 (Dropout)         (None, 7, 7, 256)         0
_________________________________________________________________
flatten_7 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_15 (Dense)             (None, 256)               3211520
_________________________________________________________________
dropout_22 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 257
=================================================================
Total params: 3,665,985
Trainable params: 3,665,985
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 150 steps, validate for 50 steps
Epoch 1/10
150/150 [==============================] - 180s 1s/step - loss: 0.7081
- accuracy: 0.4935 - val_loss: 0.6988 - val_accuracy: 0.5100
```

```
Epoch 2/10
150/150 [==============================] - 191s 1s/step - loss: 0.6893
- accuracy: 0.5427 - val_loss: 0.6982 - val_accuracy: 0.5337
Epoch 3/10
150/150 [==============================] - 191s 1s/step - loss: 0.6835
- accuracy: 0.5719 - val_loss: 0.6698 - val_accuracy: 0.5831
Epoch 4/10
150/150 [==============================] - 184s 1s/step - loss: 0.6710
- accuracy: 0.5917 - val_loss: 0.6650 - val_accuracy: 0.5869
Epoch 5/10
150/150 [==============================] - 185s 1s/step - loss: 0.6606
- accuracy: 0.6146 - val_loss: 0.6491 - val_accuracy: 0.6175
Epoch 6/10
150/150 [==============================] - 182s 1s/step - loss: 0.6437
- accuracy: 0.6381 - val_loss: 0.6170 - val_accuracy: 0.6700
Epoch 7/10
150/150 [==============================] - 181s 1s/step - loss: 0.6258
- accuracy: 0.6477 - val_loss: 0.6058 - val_accuracy: 0.6837
Epoch 8/10
150/150 [==============================] - 195s 1s/step - loss: 0.6087
- accuracy: 0.6810 - val_loss: 0.6000 - val_accuracy: 0.6825
Epoch 9/10
150/150 [==============================] - 188s 1s/step - loss: 0.6072
- accuracy: 0.6640 - val_loss: 0.6122 - val_accuracy: 0.6687
Epoch 10/10
150/150 [==============================] - 195s 1s/step - loss: 0.5962
- accuracy: 0.6831 - val_loss: 0.5919 - val_accuracy: 0.6869
```

In [41]:

```
plot_history([('aug model', aug_model_history)], key='loss')
```



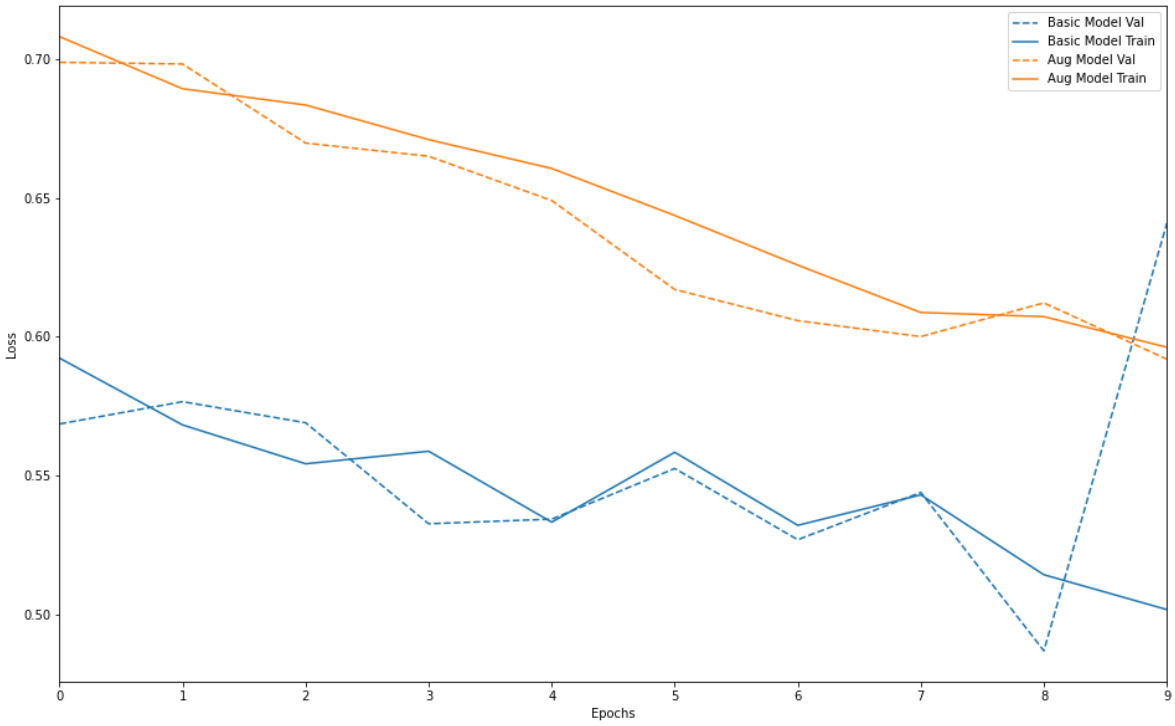
Сравним работу моделей на тестовых данных.

In [48]:

```
plot_history([('basic model', basic_model_history), ('aug model', aug_model_history

basic_test_loss, basic_test_acc = basic_model.evaluate(test_iterator)
aug_test_loss, aug_test_acc = aug_model.evaluate(test_iterator)

print('\nТочность на проверочных данных модели без аугментации:', basic_test_acc)
print('\nТочность на проверочных данных модели с аугментацией:', aug_test_acc)
```




```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
157/157 [==============================] - 42s 270ms/step - loss: 0.53
67 - accuracy: 0.7802
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
157/157 [==============================] - 44s 283ms/step - loss: 0.54
46 - accuracy: 0.7319

Точность на проверочных данных модели без аугментации: 0.780156

Точность на проверочных данных модели с аугментацией: 0.7319464
```

Задание 4. Поэкспериментируйте с готовыми нейронными сетями (например, AlexNet, VGG16, Inception и т.п.), применив передаточное обучение. Как это повлияло на качество классификатора?

In [45]:

```
pre_trained_model = keras.applications.VGG19(input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3

for i, layer in enumerate(pre_trained_model.layers):
    if i <= 42:
        layer.trainable = False
    else:
        layer.trainable = True

last_layer = pre_trained_model.get_layer('block5_pool')
last_output = last_layer.output

model = keras.layers.GlobalAveragePooling2D()(last_output)
model = keras.layers.Dense(512, activation='relu')(model)
model = keras.layers.Dropout(0.5)(model)
model = keras.layers.Dense(1, activation='sigmoid')(model)

vgg_model = keras.models.Model(pre_trained_model.input, model)

vgg_model.compile(loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy']

vgg_model.summary()
```

```
Downloading data from https://github.com/fchollet/deep-learning-model
s/releases/download/v0.1/vgg19_weights_tf_dim_ordering_tf_kernels_noto
p.h5 (https://github.com/fchollet/deep-learning-models/releases/downlo
ad/v0.1/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5)
80142336/80134624 [==============================] - 99s 1us/step
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
```

```
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense_18 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_23 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 513
=================================================================
Total params: 20,287,553
Trainable params: 263,169
Non-trainable params: 20,024,384
_________________________________________________________________
```

In [46]:

```
vgg_model_history = vgg_model.fit(aug_train_iterator, steps_per_epoch=150, epochs=1
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 150 steps, validate for 50 steps
Epoch 1/10
150/150 [==============================] - 966s 6s/step - loss: 0.6892
- accuracy: 0.5640 - val_loss: 0.6315 - val_accuracy: 0.6938
Epoch 2/10
150/150 [==============================] - 896s 6s/step - loss: 0.6401
- accuracy: 0.6300 - val_loss: 0.6018 - val_accuracy: 0.7181
Epoch 3/10
150/150 [==============================] - 1219s 8s/step - loss: 0.601
5 - accuracy: 0.6852 - val_loss: 0.5811 - val_accuracy: 0.7281
Epoch 4/10
150/150 [==============================] - 891s 6s/step - loss: 0.5863
- accuracy: 0.7008 - val_loss: 0.5783 - val_accuracy: 0.6812
Epoch 5/10
150/150 [==============================] - 1736s 12s/step - loss: 0.57
45 - accuracy: 0.7096 - val_loss: 0.5488 - val_accuracy: 0.7306
Epoch 6/10
150/150 [==============================] - 1274s 8s/step - loss: 0.554
9 - accuracy: 0.7233 - val_loss: 0.5489 - val_accuracy: 0.7325
Epoch 7/10
150/150 [==============================] - 1542s 10s/step - loss: 0.53
59 - accuracy: 0.7344 - val_loss: 0.5318 - val_accuracy: 0.7462
Epoch 8/10
150/150 [==============================] - 1796s 12s/step - loss: 0.54
12 - accuracy: 0.7266 - val_loss: 0.5376 - val_accuracy: 0.7156
Epoch 9/10
150/150 [==============================] - 1335s 9s/step - loss: 0.519
7 - accuracy: 0.7481 - val_loss: 0.5217 - val_accuracy: 0.7394
Epoch 10/10
150/150 [==============================] - 1223s 8s/step - loss: 0.520
9 - accuracy: 0.7437 - val_loss: 0.4975 - val_accuracy: 0.7688
```

In [49]:

```
plot_history([('basic model', basic_model_history), ('aug model', aug_model_history
```