glide支持gif动图，支持缩略图，支持activity生命周期的集成。

glide库比Picasso毕加索大很多。

Picasso是下载图片后，将完整大小的图片缓存到本地，用的时候再resize。

glide是下下载图片，然后改变图片的大小适应imageview要求，再缓存到本地；如果是不同的imageview要求，就会有多份缓存。

所以glide是使用适应后的，内存会小；再次使用时，加载的速度会快。

（问题：还不够清晰？）

# 1、Picasso 和Glide比较

## 1.1 picasso

picasso是Square公司开源的一个Android图形缓存库,不仅实现了图片异步加载的功能，还解决了android中加载图片时需要解决的一些常见问题：

在adapter中需要取消已经不在视野范围的ImageView图片资源的加载，否则会导致图片错位，Picasso已经解决了这个问题；

使用复杂的图片压缩转换来尽可能的减少内存消耗；

自带内存和硬盘二级缓存功能。

## 1.2 Glide

Glide 是一个高效、开源、 Android设备上的媒体管理框架，Glide具有获取、解码和展示视频剧照、图片、动画等功能，它还有灵活的API，这些API使开发者能够将Glide应用在几乎任何网络协议栈里。创建Glide的主要目的有两个，一个是实现平滑的图片列表滚动效果（滚动流畅），另一个是支持远程图片的获取、大小调整和展示。有以下特点：

GIF动画的解码：通过调用Glide.with(context).load(“图片路径“方法，GIF动画图片可以自动显示为动画效果。如果想有更多的控制，还可以使用Glide.with(context).load(“图片路径“).asBitmap()方法加载静态图片,使用Glide.with(context).load(“图片路径“).asGif()方法加载动画图片。

本地视频剧照的解码：通过调用Glide.with(context).load(“图片路径“)方法，Glide能够支持Android设备中的所有视频剧照的加载和展示。

缩略图的支持：为了减少在同一个view组件里同时加载多张图片的时间，可以调Glide.with(context).load(“图片路径“).thumbnail(“缩略比例“).into(“view组件“)方法加载一个缩略图，还可以控制thumbnail()中的参数的大小，以控制显示不同比例大小的缩略图。

Activity生命周期的集成：当Activity暂停和重启时，Glide能够做到智能的暂停和重新开始请求，并且当Android设备的连接状态变化时，所有失败的请求能够自动重新请求。

转码的支持：Glide的toBytes() 和transcode() 两个方法可以用来获取、解码和变换背景图片，并且transcode() 方法还能够改变图片的样式。

动画的支持：新增支持图片的淡入淡出动画效果（调用crossFade()方法）和查看动画的属性的功能。

OkHttp和Volley的支持：默认选择HttpUrlConnection作为网络协议栈，还可以选择OkHttp和Volley作为网络协议栈。

其他功能：如在图片加载过程中，使用Drawables对象作为占位符、图片请求的优化、图片的宽度和高度可重新设定、缩略图和原图的缓存等功能。

## 1.3 总结：

Picasso所能实现的功能，Glide都能做，只是所需的设置不同。但是Picasso体积比起Glide小太多如果项目中网络请求本身用的就是okhttp或者retrofit(本质还是okhttp)，那么建议用Picasso，体积会小很多。Glide的好处是大型的图片流，比如gif、Video，如果做美拍这种视频类应用，建议使用。

# 2、比较

Glide 和 Picasso 可以说是目前 Android 上最流行的图片加载库了。大部分安卓应用开发人员都有使用过这两个库在他们的开发工作中。这两个库也都确实提供了大量图片加载的功能，而且也都经过了很多应用的检验，是可靠可信的。表面看上去似乎两者工作原理很相似，但是实际上是有着很大差别的，主要体现在下面几个方面：

下载图片的方式

图片的缓存机制

加载到内存的机制

本文主要会围绕这几个方面来深入研究和对比两个库的差异，从而给开发者们提供参考。

对比的版本是 Glide v3.7.0 和 Picasso v2.5.2 的版本。

导入库到项目中

Picasso 和 Glide 都在 Jcenter 上有建立库，所以只需要简单的在添加dependency 即可

Picasso

dependencies {

compile 'com.squareup.picasso:picasso:2.5.1'

}

Glide

dependencies {

compile 'com.github.bumptech.glide:glide:3.5.2'

}

## 2.1 库的大小和方法的数量

对比两个.jar 库的大小，Glide 要比 Picasso 大很多，基本上是 Picasso 的3.5倍：

从库的大小，我们就可以预见，Glide 的方法必然是要大于Picasso 的，Picasso 的方法 总共有849个，而 Glide 的有2678个。

使用方式

如果只是简单的从一个 URL 中下载图片，然后显示到 imageView 中，那么两个库的使用方式基本相似，也都非常的简单。同时两个库也都支持动画和大小的剪切，也可以设置加载时候的预设图片等功能：

Picasso:

Picasso.with(myFragment)

.load(url)

.centerCrop()

.placeholder(R.drawable.loading\_spinner)

.into(myImageView);

Glide:

Glide.with(myFragment)

.load(url)

.centerCrop()

.placeholder(R.drawable.loading\_spinner)

.crossFade()

.into(myImageView);

但是，Glide 这里有一个非常招人喜欢的地方，就是 Glide 在设计的时候，就有 Activity 和 Fragment 的生命周期。什么意思呢？ 就是说你可以传递 Activity 或者 Fragment 的 context 给 Glide.with()， 然后 Glide 就会非常智能的同 Activity 的生命周期集成， 比如 OnResume 或者 onPause():

## 2.2 缓存大小

两个库也都支持缓存图片，都通过下载图片后，缓存到本地。但是这里对于缓存本地的机制，两个库是完全不同的做法。

Picasso 是下载图片然后缓存完整的大小到本地，比如说图片的大小是1080p的，之后如果我需要同一张图片，就会返回这张 full size 的，如果我需要resize，也是对这种 full size 的做 resize。

Glide 则是完全不一样的做法。Glide 是会先下载图片，然后改变图片的大小，以适应 imageView 的要求，然后缓存到本地。 所以如果你是下载同一张图片，但是设定两个不一样大小的 imageView, 那么Glide 实际上是会缓存两份。

换个角度来看，这里不仅仅是缓存的问题，比如一个 ImageView 要改变它的大小，PIcasso 就只需要下载一次 full size 的图片，但是 Glide 实际上就不仅仅是下载一次了，它需要去单独下载然后改变大小适配 imageView，因为对于 Glide 来讲，需要缓存不同大小的同一张图片。

从这点来看，似乎 Glide 的这种设计很有问题了？当然不是，这种做法也会带来一定的好处，在下面的memory中就会展示。

内存使用

Glide 默认是用的 RGB\_555 的设定，PIcasso 则是用的 ARGB \_8888的设定。为了公平起见，我这里修改了 GlideModule，让 Glide 也使用 ARGB \_8888的格式，做法也很简单，新建一个类然后继承 GlideModule

<meta-data android:name="example.com.myanimation.GlideConfiguration" android:value="GlideModule"/>

public class GlideConfiguration implements GlideModule {

@Override

public void applyOptions(Context context, GlideBuilder builder) {

// Apply options to the builder here.

builder.setDecodeFormat(DecodeFormat.PREFER\_ARGB\_8888);

}

@Override

public void registerComponents(Context context, Glide glide) {

// register ModelLoaders here.

}

}

可以看出 Glide (大约8MB) 要比 Picasso (13 MB) 的少不少。其实这里就体现出了之前提到的 Glide 缓存的方式问题。Picasso 是缓存的 full size 图片的大小，而 Glide 是加载已经改变大小后的图片，显而易见使用的memory会小很多。 这样可以减少 OutOfMemoryError 的可能性。

## 2.3 加载图片的时间

这里先说明下，当尝试加载一个图片的时候，两个库都会采用先从缓存中读取，如果缓存中没有，再去下载的做法。

实际试验中，Picasso 会比 Glide 快一点。猜测可能的原因还是因为之前讲到的缓存机制导致，因为Picasso 是直接把图加载到内存中，而 Glide 则需要改变图片大小再加载到内存中去。这个应该是会耗费一定的时间。

但是，当加载图片从内存中的时候，Glide 则比 Picasso 要快。其原理还是因为缓存机制的区别。因为Picasso 从缓存中拿到的图片，还要先去 resize 后，然后设定给 imageView，但是 Glide 则不需要这样。

## 2.4 其他功能的对比

GIF 支持：Glide 支持 GIF。 对于加载 GIF 来说，Glide 只需要简单使用 Glide.with(...).load(...)。 但是 Picasso 是不支持的，因此如果你的应用中是需要加载 GIF 的话，那就只能用 Glide 了。

灵活性：Glide 提供了非常多的配置，你可以非常灵活的根据你的需求来客制化，从而缩减 Glide 库的大小等。

# 3、结论

正所谓人无完人，经过一番对比，Picasso 和 Glide 各有千秋，那么到底我们应该用哪个库呢？这个还是回到应用的需求来看，比如你想要你的 app 小一些，没有那么多的额外功能，那么 Picasso 是你的首选。反之，比如你的应用中需要加载 GIF，或者对于内存的大小比较在意，那么 Glide 应该是不错的选择。

实际上，就我个人来看，Glide 要略优于 Picasso. 特别是支持 GIF，这个杀手锏啊！